Journal of Korea Multimedia Society Vol. 12, No. 2, February 2009(pp. 290-299)

동적 상태 진화 신경망에 기반한 팀 에이전트의 진화

김향화[†], 장동헌[††], 김태용[†††]

요 약

진화하는 인공신경망은 인공지능분야와 게임 NPC의 지능 설계 분야에서 새롭게 각광을 받고 있다. 하지만 진화하는 인공신경망을 이용하여 게임 NPC의 지능을 설계할 때 인공신경망의 구조가 복잡함에 따라 진화와 평가에 필요한 연산량이 크며 또한 적절한 적합도 함수를 설계하지 못하면 지능적인 NPC를 설계할 수 없는 등의 문제점을 가지고 있다. 본 논문에서는 이러한 문제들을 해결하고자 동적 상태 진화 인공신경망을 제안한다. 동적 상태 진화 인공신경망은 전통적인 진화하는 인공신경망 알고리즘에 기반하여 진화 과정에서 신경망의 신경세포들 사이의 시냅스를 제거(disabled) 하거나 고정(fixed)시키는 방법을 통하여 진화와 평가과정에 소모되는 연산량을 줄이는 알고리즘이다. 본 논문은 Darwin Platform 을 테스트 베드로 축구 게임 NPC의 지능 설계를 통하여 제안하는 방법의 유용성을 검증한다.

Evolving Team-Agent Based on Dynamic State Evolutionary Artificial Neural Networks

Xianghua Jin[†], DongHeon Jang[††], TaeYong Kim[†††]

ABSTRACT

Evolutionary Artificial Neural Networks (EANNs) has been highly effective in Artificial Intelligence (AI) and in training NPCs in video games. When EANNs is applied to design game NPCs' smart AI which can make the game more interesting, there always comes two inportant problems: the more complex situation NPCs are in, the more complex structure of neural networks needed which leads to large operation cost. In this paper, the Dynamic State Evolutionary Neural Networks (DSENNs) is proposed based on EANNs which deletes or fixes the connection of the neurons to reduce the operation cost in evolution and evaluation process. Darwin Platform is chosen as our test bed to show its efficiency: Darwin offers the competitive team game playing behaviors by teams of virtual football game players.

Key words: Game AI(게임 인공지능), Neural Networks(신경망), Genetic Algorithm(유전자 알고리즘), Evolutionary Artificial Neural Networks(진화 인공 신경망)

1. 서 론

게임 산업은 스토리, 그래픽, 영상, 사운드 등의 다양한 매체와 수학, 물리학, 컴퓨터 그래픽 및 프로그래밍 등의 기술을 결합한 문화 기술의 복합체로서 산업적, 문화적으로 빠른 속도로 성장하고 있다. 과

※ 교신저자(Corresponding Author) : 김태용, 주소 : 서울특별시 동작구 흑석동 아트센터 지하1층 106호(608-743), 전화 : 02)812-5717, FAX : 02)814-5404, E-mail : kimty@cau.ac.kr
접수일 : 2008년 9월 7일, 완료일 : 2008년 11월 24일
[†] 준회원, 중앙대학교 첨단영상대학원 (E-mail : hyanghwa_kim@naver.com)
[††] 중앙대학교 첨단영상대학원 (E-mail : tellamon@gmail.com)
[†††] 정회원, 중앙대학교 첨단영상대학원
※ 본 연구는 ITRC(Information Technology Research Center)와 서울시 산학연 협력사업의 지원으로 수행되었음.

거에는 제한된 컴퓨팅 리소스로 인해 많은 제약 사항이 있었지만, 현재는 향상된 하드웨어로 인하여 사실적인 그래픽 표현이 가능하게 되었으며, 앞으로의 경향은 현재보다 뛰어난 그래픽 및 사운드의 표현과 더불어 보다 사실적인 게임 환경 구축을 위하여 최첨단의 기술들이 도입될 것이다. 그중 게임의 인공지능은 게임성과 몰입감을 향상시키므로 게임의 본질과도 밀접한 연관이 있으며, 차세대 게임 기술의 핵심으로 부각되고 있다. 실제로 그래픽이나 사운드의 기능이 일정한 수준에 도달하자 게이머들은 보다 자연스럽고 재미있는 게임을 요구하게 되었으며, 이로 인하여 1990년대 후반부터 인공지능 기술이 게임에서 중요한 역할을 하기 시작하였다[1].

현재 많은 게임에서 사용되고 있는 인공지능 기법들로는 규칙 기반 시스템(Rule-based System), 유한상태 머신(Finite-state Machine), 인공신경망(Artificial Neural Networks), 유전자 알고리즘(Genetic Algorithm)등이 있다. 그중 규칙 기반 시스템과 유한상태 머신과 같은 기법들은 게임 상황이 그리 복잡하지 않은 경우에 적합하며 일단 게임 환경이 복잡하게 되면 많은 연산량과 코딩량을 필요로 하기 때문에 적합하지 않다. 또한 하이 레벨의 인공지능 기법으로 불리우고 있는 인공신경망 기법과 유전자 알고리즘도 각자의 장점과 단점을 가지고 있으며 축구 게임과 같은 상태가 복잡하면서도 또한 학습된 데이터가 없는 경우에는 사용하기가 적합하지 않다.

인공신경망, 유전자 알고리즘 등 분야의 신속한 발전과 더불어 연구자들은 인공신경망과 유전자 알고리즘을 결합하는 시도를 하게 되었으며 진화하는 인공 신경망 알고리즘을 개발하였다[2-6]. 진화하는 인공 신경망(Evolutionary Artificial Neural Networks-EANNs) 은 인공 신경망 구조를 가진 개체를 유전자로 코딩을 하여 진화하는 과정을 통하여 상황에 적합한 개체를 얻어내는 과정으로써 현재 많은 게임에서의 NPC들의 온라인/오프라인 학습에서 많이 쓰이고 있는 인공지능 기법중의 하나이다[7].

일반적으로 게임에서 NPC의 지능을 설계하고 구현을 함에 있어서 해당 게임의 환경의 변화와 이왕의 경험 등 지식에 근거하여 NPC 가 유저의 행동에 대한 반응을 설계하게 된다. 게임 환경이 복잡할수록 NPC의 지능설계도 어려워지게 된다. 진화하는 인공신경망 기법을 통하여 NPC의 지능을 설계함에 있어서도 게임환경의 복잡성이 증가하면 인공 신경망의 입출력층의 뉴런의 개수가 증가하게 되며 따라서 진화에 필요한 연산량도 증가하게 된다.

본 논문에서는 위에서 서술한 현재 게임 NPC 지능 설계에 존재하는 문제에 기반을 두고 기존의 진화하는 인공신경망에 기반한 동적 상태 진화 인공신경망(Dynamic State Evolutionary Neural Networks-DSENNs)을 제안하고 이 기법을 경쟁성 축구 팀 게임에 접목 시켜 축구 게임 환경에 적합한 지능성 NPC를 구현 하고자 한다.

본 논문에서는 인공지능 기반 게임 플랫폼인 Darwin[8]을 테스트 베드로 사용하였다. Darwin Platform은 "Ergonomics Game Intelligence" 프로젝트에서 개발된 인공신경망, 유전자 알고리즘 ,유한상태 머신 등의 기본적인 인공지능 기법을 제공하는 게임 플랫폼이다. Darwin Platform은 AI-레벨, 게임-레벨 2가지 영역으로 구분되어 있다. 게임 개발자는 Darwin에서 제공하는 템플릿을 사용하여 DLL을 제작한다.

본 논문에서는 Darwin Platform이 제공하는 홍팀과 청팀 5:5 로 구성된 축구 게임에서의 홍팀의 공격수 2명을 진화하고자 한다. 축구 게임중 청팀은 유한상태 머신으로 프리 프로그래밍 되어 홍팀의 공격수들의 진화를 돕는 역할을 하며 홍팀의 공격수들은 NEPlayer라는 이름을 가지고 제안하는 알고리즘인 동적 상태 진화 인공신경망알고리즘으로 진화를 하게 된다.본 실험에서는Darwin AILib 를 사용하여 DLL을 구현함에 있어서 Darwin AILib 에 있는 Neuro-Evolution(EANN) 알고리즘 을 사용한 축구 전략 선정 시스템을 사용하여 오프라인에서 트레이닝된 에이전트를 실시간으로 로딩하여 게임에 투여할 수 있게 하였다.

본 논문의 주요 구성은 다음과 같다.

2장에서 기존 연구에 대한 설명을 하고 3장에서 본 논문에서 제안하는 동적 상태 진화 인공신경망과 이 알고리즘을 축구 게임 NPC 지능 설계에 접목시킬 경우 NPC가 가지는 신경망 구조와 그 진화 과정에 필요한 적합도 함수에 대한 설계에 대하여 설명한다. 본 논문의 4장에서는 실험 및 그 결과에 대하여 분석을 하고 5장에서 결론을 짓는다.

2. 진화하는 인공신경망 기법을 게임 NPC의 지능설계에 사용한 연구 동향

진화하는 인공신경망은 인공신경망을 유전자로 코딩을 하여 유전 알고리즘의 진화과정을 통하여 최적의 신경망을 찾아내는 알고리즘으로써 그 복잡성은 어떤 유형의 신경망 구조를 선택할 것인가에 의하여 좌우된다. 일반적으로 신경망 구조의 유형은 전문가적인 지식에 근거하여 선택할 수 있다.사전에 트레이닝된 데이터를 필요로 하는 경우에는 역전파 신경망 구조를 사용 하지만 강화학습 문제와 같은 사전에 트레이닝된 데이터가 없는 경우는 일반적으로 전방향 신경망 구조만 가지고 진화를 시킨다[5].

본 논문에서 해결하고자 하는 축구 게임 NPC의 지능설계와 같은 경우는 스스로 학습과정을 통하여 진화를 하게 되는 경우로써 강화학습의 일련의 문제를 해결하는데 적합한 전방향 인공신경망의 진화 구조를 사용하고자 한다. 본 장에서는 전방향 인공신경망의 진화기법을 게임 NPC의 지능설계에 사용한 몇 가지 사례에 대하여 분석한다.

2.1 Enforced Sub-Populations 기법을 캡쳐(Capture) 게임에 사용

Enforced Sub-Populations (ESP) [9]은 진화하는 인공신경망기법중의 하나로써 Symbiotic, Adaptive Neuro-Evolution (SANE) [10]에 기존한다. SANE 기법과 ESP는 모두 신경망의 뉴런들을 진화하는 방법을 통하여 최적화의 신경망 구조를 얻어내는 기법들이다. 이 기법들은 각각의 뉴런을 하나의 유전 인자로 보고 이러한 유전 인자들의 조합을 하나의 유전자로 코딩을 하는 점에서 전통적인 진화하는 인공신경망 기법과 틀리다. ESP는 각각의 독립된 그룹 안에서 하나씩 뉴런을 뽑아내어 그 뉴런들로 은닉층을 구성하고 해결하려는 문제에 인공신경망을 실행하고 적합도의 값을 매개의 뉴런에 평균하는 방법을 사용한다. SANE는 하나의 전체 그룹 안에서 몇 개의 뉴런을 뽑아내어 하나의 은닉층을 구성하지만 ESP는 같은 위치에 있는 은닉층 뉴런들의 집합을 하나의 독립된 그룹으로 본다는 점에서 더 진보적이다.

논문[11]에서는 위에서 서술한 ESP기법을 1대 다, 혹은 3대 다의 캡쳐 게임 NPC 의 지능설계에 사용하였다.

2.2 NeuroEvolution of Augmenting Topologies 기법을 슈팅 게임에 사용

NeuroEvolution of Augmenting Topologies (NEAT) [12]는 또 하나의 진화하는 인공신경망 기법으로써 실시간 슈팅 게임 NPC의 지능설계에 사용되었다.NEAT는 작은 구조의 인공신경망으로부터 시작하여 세대가 증가함에 따라 구조를 증가해 나가는 방법을 사용한다. 이러한 기법은 실시간으로 NPC들이 실시간으로 죽고 되살아나는 과정을 반복하는 슈팅게임에 적합하다. 이러한 게임에서 하나의 NPC는 하나의 인공신경망 구조를 가지고 있는 개체로써 실시간 게임 환경에서의 일정한 반경 안에 있는 같은 팀 팀원과 적들의 수, 총알의 있고 없음, 주변에 숨을 수 있는 벽의 있고 없음 등의 정보들이 입력값으로 되고 NPC가 가야 할 방향 , 총을 쏴야 할지 여부 등의 행동들이 그의 출력값으로 된다.

NEAT기법을 사용하여 진화된 NPC 는 슈팅 게임에서 총알을 잘 피해나가면서 적들을 죽일 수 있는 지능적인 개체로 되었다.

위에서 설명한 바와 같이 진화하는 인공신경망 기법을 사용하여 진화된 NPC 의 자아학습 시스템은 NPC로 하여금 자주적, 지능적인 개체로 되게 하였으며 많은 분야에서 좋은 성적을 거두었다. 하지만 이러한 기법들은 간단한 신경망 구조의 진화와 간단한 지능을 얻어내는데 국한되어 있으며 축구 게임과 같은 복잡한 게임 환경에 적합한 NPC의 지능 설계에는 적합하지 않다. 본 논문의 다음 장에서는 복잡한 신경망 구조의 진화과정에서 시냅스의 상태를 변화시키는 과정을 통하여 연산속도를 줄일 수 있는 동적 상태 진화 인공신경망 기법을 제안하는 동시에 축구 게임환경에 적합한 NPC 지능 설계의 전반과정을 소개한다.

3. 동적 상태 진화 인공신경망

입력층으로부터 출력층 방향으로의 모든 뉴런들이 다 연결되어 있는 인공신경망의 시냅스들 가운데는 신경망의 전반 구조에서의 작용이 아주 미소하거나 또는 작용을 전혀 하지 못하는 불필요한 시냅스들이 존재한다[5].이러한 시냅스 가중치의 진화과정과 평가과정에서의 연산량을 줄이고자 본 논문에서는 진화하는 과정에서 불필요하다고 판단되는 뉴런들

사이의 시냅스의 연결 상태를 변화시키는 방법을 통하여 진화과정과 평가과정에 필요한 연산량을 대폭 줄이는 단계적 시냅스 제거 진화 인공 신경망 알고리즘을 제안한다. 또한 본 장에서는 제안하는 알고리즘으로 진화하는 축구 팀 게임 NPC를 NEPlayer라 하고 그 신경망 구조와 적합도 함수에 대하여서도 설명한다.

3.1 동적 상태 진화 인공신경망

동적 상태 진화 인공신경망 알고리즘은 전통적인 진화하는 인공신경망에 기반을 하며 그 주요 과정은 다음과 같다.

Step 1: **초기화.** -1~1 사이의 랜덤 가중치들로 구성된 full-linked 인공신경망들로 초기의 인구수를 구성 한다. 본 논문의 실험에 사용되는 인공신경망은 시냅스가 190개를 초과하는 큰 구조의 신경망으로써 전통적인 2진 코딩 유전 알고리즘을 사용하면 유전자의 길이의 과대를 초래하기 때문에 실수코딩 방법을 사용하며 또한 시냅스의 상태 변화를 용이하게 하기 위하여 각각의 시냅스는 하나의 유전인자로써 가중치의 크기와 연결 상태(connected, fixed 혹은 disabled) 등 두 가지 정보를 가지며 이러한 시냅스들의 조합들로 하나의 전체 신경망 구조를 나타낸다. 그 코딩 방법은 그림 1에서 보여지는바와 같다.

Step 2: **평가.** 각각의 신경망구조를 가지는 NPC를 게임에 실행시켜 적합도를 부여하는 방식으로 각각의 신경망을 평가한다. 적합도 함수에 대한 상세한 설명한 다음절에서 한다.

Step 3: **진화 과정.** 룰렛 휠 선택 방법으로 부모 신경망을 선택하고 해당 유전 연산을 진행하여 자식을 생성 한다. 위에서 설명한바와 같이 본 논문에서는 실수코딩 방법을 사용하였으며 해당 유전연산은 참고문헌[13,14]에서 제안한 탐색 공간이 큰 문제에서 최적의 해를 구하는데 강건한 유전 연산 방법인 Simplified differential operator(SADE), Mutation, Local Mutation 등의 방법을 사용한다. 각각의 유전연산은 다음과 같다.

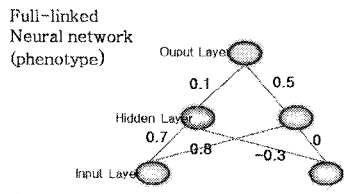
1) Simplified differential operator: $C_i(t)$가 t번째 세대에서의 i번재 유전자라고 할 때,

$$C_i(t) = (c_{i1}(t), c_{i2}(t), .. c_{in}(t)) \tag{3-1}$$

여기서 n은 유전인자 즉 하나의 신경망에 있는 시냅스의 전체 개수를 나타낸다. 이때 SADE 연산은 다음과 같다:

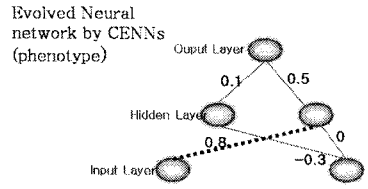
$$c_{ij}(t+1) = c_{pj}(t) + \tau_{CR}*(c_{qj}(t) - c_{rj}(t)) \tag{3-2}$$

c_{pj}, c_{qj} 와 c_{rj} 는 랜덤하게 선택한 세 개의 유전자의



Genotype

status	connected	connected	connected	connected	connected	connected
weight	0.7	0.8	-0.3	0	0.1	0.5



Genotype

status	disabled	fixed	connected	connected	connected	connected
weight	0	0.8	-0.3	0	0.1	0.5

그림 1. 유전코딩 방법

j번째 값이며 τ_{CR}은 교배율이다.

2) Mutation: Mutation연산은 다음과 같다.

$$C_k(t+1) = C_i(t) + \tau_{MR} * (C_{RP} - C_i(t)) \quad (3\text{-}3)$$

여기서 C_{RP}는 새롭게 만들어진 유전자이고 τ_{MR}은 변이율이다.

3) Local mutation: 정해진 유전자가 이 연산을 수행할 때, 그 유전자의 모든 유전인자의 값은 주어진 작은 값에 곱해진다.

이상의 모든 연산은 시냅스의 상태가 "connected"일 때만이 수행 된다. 알고리즘으로 나타내면 다음과 같다.

if $status = disabled$
$\{SADE = Mutation = Local\,Mutation = false,$
$c_{ij}(t+1) = c_{ij}(t) = 0;\}$

if $status = fixed$
$\{SADE = Mutation = Local\,Mutation = false,$
$c_{ij}(t+1) = c_{ij}(t);\}$

$else$ if $status = connected$
$DO\ \{SADE = Mutation = Local\,Mutation;\}$

Step 4: **재평가.** 생성된 자식들은 평가를 받고 적합도 함수를 부여받는다.

Step 5: **선택.** 랜덤하게 두 개체를 선택하고 적합도가 낮은 개체를 버린다.

Step 6: **시냅스의 상태 결정.** 최고 적합도의 값이 주어진 제한 적합도의 값을 넘은 개체가 나타난 세대부터 시작하여 각각의 간선의 상태(connected, disabled, fixed)를 결정한다.

같은 세대 안에서의 신경망들 중에서 같은 위치에 있는 시냅스중 적합도가 제일 높은 개체와 적합도가 제일 낮은 개체에서의 가중치의 절대 값들이 주어진 제한 가중치의 값보다 작고 또한 그 차이값이 작으면 그 시냅스는 적합도의 값의 높고 낮음을 불문하고 신경망 구조에서의 작용이 미소하다고 판단하고 그 상태를 "disabled"로 하고 그 시냅스는 진화과정과 평가과정에 전부 참가시키지 않는다.

알고리즘으로 나타내면 다음과 같다.

if $w_{ij}^{best}(t) < \delta_w^{best},\ w_{ij}^{best}(t) - w_{ij}^{worst}(t) < \delta_w^{diff}$
$\{status = disabled;\}$

if $w_{ij}^{best}(t) < \delta_w^{best},\ w_{ij}^{best}(t) - w_{ij}^{best}(t-n) < \delta_w^{diff}$
$\{status = disabled;\}$

if $w_{ij}^{best}(t) > \delta_w^{best},\ w_{ij}^{best}(t) - w_{ij}^{best}(t-n) < \delta_w^{diff}$
$\{status = fixed;\}$

즉 한 세대 안에서의 적합도가 제일 높은 신경망과 몇 세대 전의 적합도가 제일 높은 신경망을 비교하였을 때, 같은 위치에 있는 시냅스의 가중치의 절대값이 주어진 제한 가중치의 값보다 작고 또한 둘 사이의 차이값이 주어진 제한값보다 작으면 그 위치에 있는 시냅스도 세대가 지남에도 신경망 안에서의 작용이 작다고 판단하고 그 상태를"disabled"로 하고 그 시냅스는 진화과정과 평가과정에 전부 참가시키지 않는다.

같은 세대 안에서의 신경망들 중에서 같은 위치에 있는 시냅스중 적합도가 제일 높은 개체와 적합도가 두 번째로 높은 개체에서의 가중치의 값들이 주어진 제한 가중치의 값보다 크고 또한 몇 세대 이전의 적합도가 제일 높았던 개체와 비교 하였을 때 현저한 변화가 없으면 신경망 구조에서 차지하는 비중은 크지만 더 이상 진화의 여지가 없다고 판단하고 그 상태를 "fixed"로 하고 그 시냅스는 진화과정에 참가하지 않고 평가과정에만 참가한다.

Step 7: **멈춤 조건 판단.** 최종 적합도의 상한 값을 넘는 개체가 나타나고 또한 더 이상 좋은 개체가 나타나지 않을 때까지 2-6을 반복한다.

3.2 NEPlayer 의 구조

본 실험은 테스트 대상(OurPlayer) 그룹과 비교 대상(EnemyPlayer) 그룹 player들의 위치 좌표, 공의 좌표 값을 Neural Network의 입력 값으로 넘겨받고, 현재 NEPlayer가 공을 소유하고 있는지의 여부에 따라 그에 상응하는 출력 값 즉 행동을 얻게 된다. 진화하는 동안 NEPlayer 는 다음과 같은 두 가지 경우의 서로 다른 신경망 구조를 가질 수 있다.

1) NEPlayer 가 공을 가지고 있지 않을 때 : NEPlayer는 테스트 대상 player와 테스트 대상 그룹의 팀원들 사이의 상대 좌표 값, 축구공이 AI Soccer에 대한 상대 좌표 값, 비교 대상 팀원들 사이의 상대 좌표 값 등을 입력 값으로 받게 되며, 받아들여진 입력 값은 신경망 구조의 은닉층과 출력층에 있는 가중치들에 의한 연산을 통하여 최종적으로 출력 층에서 Sigmoid 함수에 의해 출력된다. 이때의 출력 값은 NEPlayer가 공을 가지고 있지 않을 때 어느 방향으로 움직여야 할지를 결정한다. 출력된 세 개의 값은 000~111조합으로써 8개의 방향을 결정한다(그림 2).

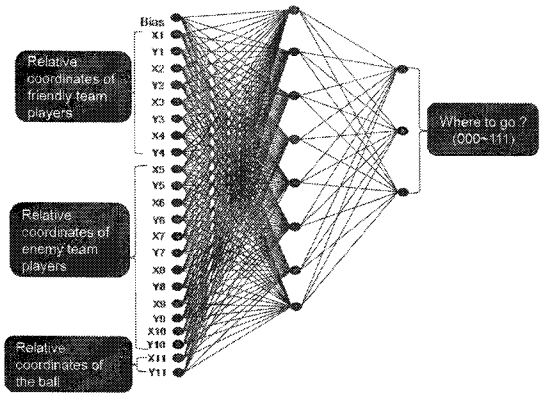


그림 2. NEPlayer 가 공을 가지고 있지 않을 경우의 신경망 구조도

각각의 출력값에 대응하는 해석은 표 1에서 보여지는 바와 같다.

2) NEPlayer가 공을 가지고 있을 때 : NEPlayer는 테스트 대상 플레이어와 테스트 대상 그룹의 팀원들 사이의 상대 좌표 값, 비교 대상 팀원들 사이의 상대 좌표 값 등을 입력 값으로 받게 되며, 이 때 출력 값은 공을 가지고 있을 때 어느 NEPlayer가 어떤 방향으로 움직이며, 얼마 만큼의 Power로 공을 차야 할지를 결정하게 된다. 처음 세 개의 출력 단은 000~111조합으로써 8개의 방향을 결정하게 되며 , 4~5번째 출력 단은 00~11조합으로써 공을 차는 4가지 서로 다른 power를 결정한다(그림 3).

각각의 출력값에 대응하는 해석은 표 2에서 보여지는 바와 같다. 예를 들어 첫 두 출력단의 조합이

표 1. NEPlayer 가 공을 가지고 있지 않을 경우의 출력값의 의미

출력값	상응한 행동
000	공과 비교 대상(EnemyPlayer) 골문대 중점사이의 벡터로 이동한다.
001	제일 가까운 거리에 있는 테스트 대상(Ourl'layer) 과 공사이의 중간벡터로 이동한다.
010	두 번째로 가까운 거리에 있는 테스트 대상 과 공사이의 중간벡터로 이동한다.
011	세 번째로 가까운 거리에 있는 테스트 대상과 공사이의 중간벡터로 이동한다.
100	공이 있는 위치로 달려간다.
101	원점 기준으로 0 도 방향으로 이동한다.
110	원점 기준으로 120 도 방향으로 이동한다.
111	원점 기준으로 240 도 방향으로 이동한다.

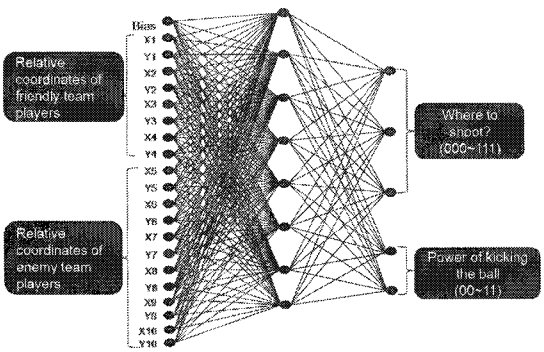


그림 3. NEPlayer 가 공을 가지고 있을 경우의 신경망 구조도

표 2. NEPlayer 가 공을 가지고 있을 경우의 출력값의 의미

출력값	상응한 행동
000	제일 가까운 테스트 대상(OurPlayer)한테 공을 찬다.
001	두 번째로 가까운 테스트 대상한테 공을 찬다.
010	세 번째로 가까운 테스트 대상한테 공을 찬다.
011	제일 가까운 비교 대상 (EnemyPlayer)과 제일 멀리 있는 비교 대상사이의 중간 벡터로 공을 찬다.
100	두 번째로 가까운 비교 대상과 제일 멀리 있는 비교 대상사이의 중간 벡터로 공을 찬다.
101	원점 기준으로 0 도 방향으로 공을 찬다.
110	원점 기준으로 120 도 방향으로 공을 찬다.
111	원점 기준으로 240 도 방향으로 공을 찬다.

00이고 3~5번째 출력단의 조합이 001 이면 NEPlayer는 두 번째로 가까운 테스트 대상(OurPlayer)한테 공을 찰 수 있는 max 파워의 1/4 만큼의 파워로 공을 찬다.

3.3 NEPlayer의 적합도 함수

적합도 함수를 설계할 때에도 위에서 설명한 NEPlayer가 공을 가지고 있는지의 여부에 따라서 따로 설계한다. 또한 각각의 적합도 함수는 모두 한번의 게임이 끝났거나 혹은 일정한 시간동안 골이 나지 않을 경우 그 게임동안의 NEPlayer가 했던 행동들에 대한 평가의 누적치를 최종 적합도 값으로 매겨주도록 설계를 함으로써 NEPlayer가 골을 넣는데만 집착하지 않고 가능한 다양한 지능적인 행동을 하도록 하려는데 목적을 두었다.

1) NEPlayer가 공을 가지고 있지 않을 때 : 이 경우

적합도 함수는 각 팀이 공을 소유한 차수, 플레이동안 맴버들이 적절하게 퍼져있는 정도, NEPlayer가 적합한 위치를 찾아간 횟수 등에 기반 한다.

(1) 테스트 대상이 골을 넣었을 경우:

$$f = (N_o - N_E + 0.5*N_{Bonus} + N_{GP} - N_{BP})/T \quad (3\text{-}4)$$

(2) 비교 대상이 골을 넣었을 경우 :

$$f = -(N_o - N_E + 0.5*N_{Bonus} + N_{GP} - N_{BP})/(3000 - T_f) \quad (3\text{-}5)$$

(3) 일정한 시간 동안 골이 나지 않을 경우 :

$$f = 0.5*(N_o - N_E + 0.5*N_{Bonus} + N_{GP} - N_{BP})/T_f \quad (3\text{-}6)$$

여기서 f 는 적합도 함수, N_o와 N_E는 각각 테스트 대상 팀원이 공을 소유한 횟수와 비교대상 팀원이 공을 소유한 횟수이고, N_{Bonus}는 게임 중 각각의 팀원들 사이의 거리의 합이 일정한 상한값을 넘지 않았을 경우의 횟수이며, T는 골이 들어가는데 걸리는 시간, T_f 는 일정한 시간동안 골이 나지 않을 경우의 프레임 제한값이고 N_{GP}와 N_{BP}는 각각 NEPlayer가 한번의 게임을 하는 동안 자신의 일정한 반경안을 체크하여 좋다고 판단되는 위치에 있은 경우의 횟수와 나쁜 위치에 있은 경우의 횟수와 나쁜 위치에 있은 경우의 횟수를 나타낸다.

2) NEPlayer가 공을 가지고 있을 때 : 이 경우 적합도 함수는 플레이어가 공을 얼마나 잘 넘겼는지에 기반 한다.

(1) 테스트 대상이 골을 넣었을 경우:

$$f = (S_F \times N_F + S_E \times N_E + S_{NF} \times N_{NF} + S_{NE} \times N_{NE})/T \quad (3\text{-}7)$$

(2) 비교 대상이 골을 넣었을 경우 :

$$f = -(S_F \times N_F + S_E \times N_E + S_{NF} \times N_{NF} + S_{NE} \times N_{NE})/T \quad (3\text{-}8)$$

(3) 일정한 시간 동안 골이 나지 않을 경우 :

$$f = 0.5 \times (S_F \times N_F + S_E \times N_E + S_{NF} \times N_{NF} + S_{NE} \times N_{NE})/T_f \quad (3\text{-}9)$$

여기서 f는 적합도 함수, S_F와 N_F는 테스트 대상 팀이 공을 넘겨받았을 때의 점수 +2점과 한 게임이 끝날 때까지 이런 경우의 횟수, S_E와 N_E는 비교 대상 팀이 공을 넘겨받았을 때의 점수 -2점과 그 횟수를 나타내며, S_{NF}와 N_{NF}는 공을 누구도 받지 않았을 때 가장 가까운 거리에 있는 멤버가 테스트 대상 팀 플레이어일 경우의 점수 +1점과 그 횟수를 나타내고, S_{NE}와 N_{NE}는 공을 누구도 받지 않았을 때 가장 가까운 거리에 있는 멤버가 비교 대상 팀 플레이어일 경우의 점수 -1점과 그 횟수를 나타낸다.

NEPlayer는 이러한 적합도 함수로 평가를 받으면서 진화를 하며 진화가 끝나면 적합도가 제일 높은 개체를 저장하고 게임에 로딩하여 적합도가 제일 높았던 개체의 지능부여 여부를 체크한다. 다음 장에서는 본 논문에서의 실험 및 그 결과에 대하여 설명하고자 한다.

4. 실험 및 결과

본 절에서는 실험에 대한 해석과 그 결과에 대하여 비교 ,분석을 진행 한다.

표 3은 본 실험에서 사용한 일련의 파라미터 값이다.

제안하는 동적 상태 진화 인공신경망 알고리즘의 진화과정과 평가과정에서의 감소된 연산량을 평가하기 위하여 본 논문에서는 상대 연산량을 비교하는 방식을 선택하였다. 표 4는 4가지 서로 다른 CPU에서의 4가지 연산에 대한 상대 연산량을 보여준다[15].

Column-A 는 basic processor의 상대 연산량, Column-B는 P4 processor에서 streaming SIMD와 Prescott를 사용할 때 상대 연산량, Column-C

표 3. 실험에 사용된 파라미터의 값

Parameter Type	Value
가중치의 값의 범위	-1~1
초기 신경망 개수	60
교배율	0.2
변이율	0.5
로칼 교배율	0.25

표 4. 4가지 서로 다른 CPU에서의 상대 연산량

	A	B	C	D
Integer addition	1	1	1	1
Integer multiply	4	4	1.2	24
Floating-point addition	20	3	1	4.2
Floating-point multiply	20	5	1.2	113

는 P3 MMX processor에서의 상대 연산량이며, Column-D는 C++ compiler를 사용할 때의 P4에서의 상대 연산량이다. 본 실험은 P4 processor에서의 Visual studio 2005-VC++ 컴파일러를 사용하였기 때문에 Column-D에 있는 연산량을 선택하여 평가를 하였다.

하나의 시냅스가 평가과정과 진화과정에 참가할 때의 상대 연산량은 표 4에 근거하여 다음과 같이 쓸 수 있다.

$$\begin{aligned} cost_{evaluation} &= cost_{weight*input\ value} \\ &= 113 \end{aligned} \tag{3-10}$$

$$\begin{aligned} &cost_{evolution} \\ &= (cost_{SADE} + cost_{M} + cost_{LM}) \\ &= [(113+4.2+113+4.2) \\ &\quad +(113+4.2+113+4.2)+113]/3 \\ &= 193.93 \end{aligned} \tag{3-11}$$

또한 제안하는 동적 상태 진화 인공신경망을 사용할 때 감소되는 상대 연산량은 위에서 설명한 시냅스의 상태가 "disabled" 거나 혹은 "fixed"가 된 개수와 연관되며 전체 세대에서의 전체 감소된 연산량은 세대수와 신경망의 전체 개수와도 연관된다. 식으로 나타내면 다음과 같다.

$$\begin{aligned} &cost_{total} \\ &= \sum_{g=0}^{n_g} [cost_{evol}*(n_{disabled}+n_{fixed}) \\ &\quad +cost_{eval}*n_{disabled}]*n_{genome} \\ &= \sum_{g=0}^{n_g} [193.93*(n_{disabled}+n_{fixed}) \\ &\quad +113*n_{disabled}]*n_{genome} \end{aligned} \tag{3-12}$$

본 실험에서 동적 상태 진화 인공신경망 기법을 사용하여 축구 NPC로 진화하면서 감소된 연산량을 구할 때 3장에서 서술한 제한 상한값을 각각 0.1과 0.3으로 하고 각각의 결과를 분석하였다. 그 결과는 각각 표 5 와 표 6에서 보여준다.

두 표에서 보여지는바와 같이 가중치의 비교 상한값을 0.1, 0.3 으로 설정한 모든 경우에서 많은 연산량을 감소 할 수 있었다. 상한 값이 0.3인 경우, 더 많은 연산량을 감소 할 수 있었으며 NPC의 진화과정에서의 Full-link 된 인공신경망을 진화 할 때보다 약 1/4 정도의 시냅스의 상태를 "disabled" 혹은 "fixed"로 만듦으로써 약 1/4 정도의 연산량을 줄여줄 수 있다는 것을 알 수 있다. 또한 시냅스의 상태를 변화시켰음에도 불구하고 전반적인 진화과정에서 최적의 구조를 정확하게 얻어낼 수 있음을 알 수 있다.

표 5. 제한 상한값이 0.1인 경우 실험결과

세대수	최고 적합도	Disabled	Fixed	감소된 연산량	감소된 연산량 (누적)
1~10	1	0	0	0	0
11~20	1.49	10	2	2281725.5	2,281,725
20~30	1.65	18	3	1748584.3	24,56,5839
30~40	2.3	21	5	863709	245,622,102
40~50	2.79	23	8	789129	2,466,010,149
50~60	3.02	26	12	1196996.6	24,661,221,186
60~100	3.2	27	12	810295.2	986,499,657,759

표 6. 제한 상한값이 0.3인 경우 실험결과

세대수	최고 적합도	Disabled	Fixed	감소된 연산량	감소된 연산량 (누적)
1~10	1.3	0	0	0	0
11~20	1.65	14	2	3092020.8	3,092,021
20~30	1.7	20	7	1855411.8	32,775,620
30~40	2.2	24	9	1066282.8	328,822,483
40~50	2.65	29	11	1298856.6	3,289,523,685
50~60	3.37	36	15	1929991.8	32,897,166,838
60~100	3.4	37	15	810295.2	1,315,889,914,693

본 실험에서 설계한 적합도 함수가 축구 팀 게임에서의 지능적 NPC를 설계는데 적합한지의 여부를 판단하기 위하여 진화하기전의 NEPlayer를 FSM 으로 프로그래밍된 플레이어들과 대결을 시키고 전통적인 진화하는 인공신경망 알고리즘으로 진화된 NEPlayer를 FSM 팀과 대결을 시켰으며 또한 동적 상태 진화 인공신경망으로 진화된 NEPlayer를 FSM 팀과 각각 100전씩 5번 대결을 시키고 그 결과를 비교 분석 하였다. 그 결과는 각각 그림 4, 그림 5, 그림 6에서 보여지는바와 같다.

그림 4에서 보여지는바와 같이 진화하기 전의 NEPlayer는 게임에서 아주 낮은 득점을 기록 하였으며 게임 환경에 적응할 수 있는 지능적인 행동을 하지 않았음을 알 수 있다. 그림 5 와 그림 6은 전통적인 진화하는 인공신경망과 본 논문에서 제안하는 동적 상태 진화 인공신경망으로 트레이닝 된 NEPlayer들이 게임에서 높은 득점을 기록하였으며 축구 게임에 적합한 지능형 플레이어로 되었음을 보여준다. 또한

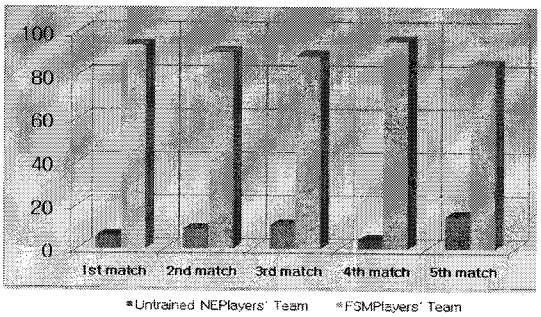


그림 4. 진화하기전의 NEPlayer팀과 FSMPlayer 팀의 대결을 한 결과

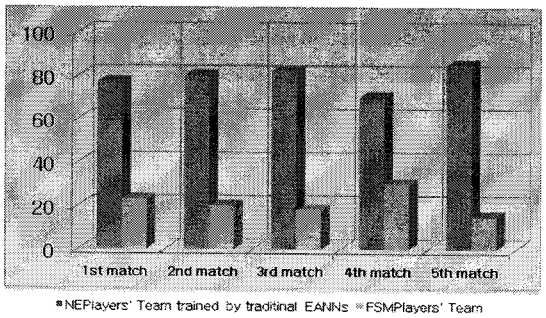


그림 5. 전통적인 진화하는 인공신경망으로 트레이닝 된 NEPlayer팀과 FSMPlayer 팀의 대결을 한 결과

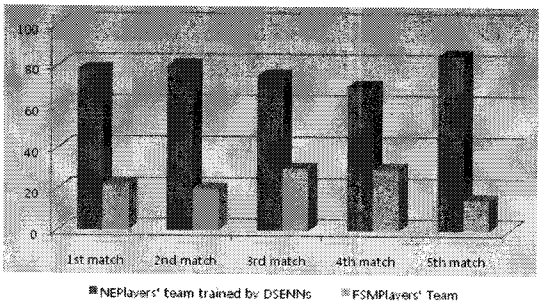


그림 5. 전통적인 진화하는 인공신경망으로 트레이닝 된 NEPlayer팀과 FSMPlayer 팀의 대결을 한 결과

NEPlayer가 한번의 게임 동안 했던 행동에 점수를 매겨주고 그 점수들을 누적하여 최종 적합도의 값에 반영 시키는 방법으로 설계한 적합도 함수는 지능적 축구 팀 게임 플레이어를 설계하는데 적합한 방법임을 알 수 있다.

이상의 실험 결과로부터 알 수 있는바 본 논문에서 제안한 알고리즘은 큰 구조의 인공 신경망을 진화할 때 진화 과정과 평가 과정에서의 연산량을 줄이는 알고리즘으로써 축구 팀 게임 NPC의 지능 설계에 적합한 방법임을 검증 하였다.

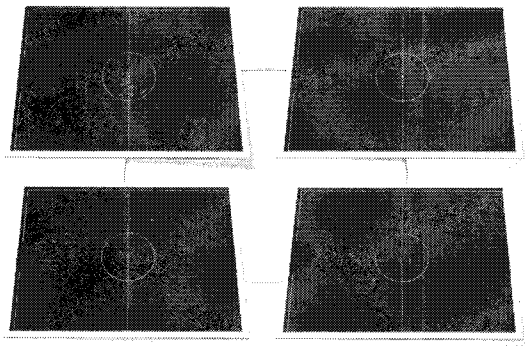


그림 7. Darwin Platform에서의 NEPlayer팀과 FSMPlayer 팀사이의 대결 스크린 샷

그림 7은 Darwin Platform에서의 NEPlayer팀(홍팀)과 FSMPlayer팀(청팀) 사이의 대결 스크린 샷을 보여준다.

5. 결 론

축구 팀 게임 환경에 적합한 NPC의 지능을 설계하기 위하여 본 논문에서는 동적 상태 진화 인공신경망 알고리즘을 제안하였다. 제안한 동적 상태 진화 인공신경망 알고리즘(DSENNs) 은 NPC의 진화과정에서의 시냅스의 연결 상태를 동적으로 변화시킴으로써 Full-link 된 인공신경망을 진화 할 때보다 약 1/4 정도의 연산량을 줄여줄 수 있다는 것을 알 수 있었다. 또한 시냅스의 상태를 변화시켰음에도 불구하고 전반적인 진화과정에서 최적의 구조를 정확하게 찾아낼 수 있다는 것을 알 수 있다. 제안한 알고리즘으로 진화된 NPC를 실제 축구 게임에 참가시켜 게임을 한 결과 비교적 지능적인 행동을 할 수 있다는 점으로부터 제안한 알고리즘의 유용성을 검증 하였다.

추후 시냅스에 대한 상세한 분석을 통하여 각각의 시냅스가 가지는 의미를 알아내어 앞으로의 게임 지능 설계에 편이성을 심어줄 수 있는데 연구를 진행하려고 한다.

참 고 문 헌

[1] Andrew Rollings and Dave Morris., *Game Architecture and Design,* 2000.

[2] Kitano, H., "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems*. pp. 4:461-476, 1990.

[3] Whitley, D., Starkweather, T., and Bogart, C., "Genetic algorithms and neural networks, Optimizing connections and connectivity," *Parallel Computing*, 14, pp. 347-361, 1990.

[4] Liu, Y., and Yao, X., "A population-based learning algorithm which learns both architectures and weights of neural networks," *Chinese Journal of Advanced Software Research*, 3(1), 1996.

[5] Yao, X., and Liu, Y., "Evolving artificial neural networks," *Proceedings of the IEEE*, Vol.87, No.9, Sept. 1999.

[6] Koza, J.R., and Rice, J.P., "Genetic generalization of both the weights and architecture for a neural network," *International Joint Conference on Neural Networks*, pp. 397-404. NY, IEEE, Vol.2, New York, 2000.

[7] Mat Buckland., *AI Techniques for game programming*, 2004.

[8] 임차섭, 김태용, 게임 NPC 지능 개발을 위한 부하분산과 그룹 행동을 지원하는 유연한 플랫폼 구조, 대한전자공학회 논문지, 제43권, 제2호, pp. 106-117, 2006년 3월.

[9] F. Gomez and R. Miikkulainen., "Incremental evolution of complex general behavior," *Adaptive Behavior* 5, pp. 317-342, 1997.

[10] Moriarty, D.E., "Symbiotic Evolution of Neural Networks in Sequential Decision Tasks," *Technical Report AI*, 1997, pp. 97-257. Austin, 1997.

[11] Chern Han Yong, Risto Miikkulainen, "Cooperative Coevolution of Multi-Agent Systems," *Technical Report AI*, pp. 01-287. University of Texas at Austin, 2001.

[12] Stanley, K.O. Bryant, B.D., Miikkulainen, R., "Real-time neuroevolution in the NERO video game," *IEEE Transactions on Evolutionary Computation*, Vol.9, No.6, pp. 653-668, Dec. 2005.

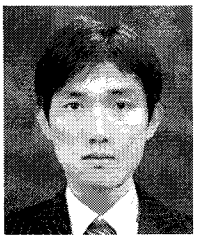
[13] Hrstka O, Kuc˘erova´, "A Search for optimization method on multidimensional real domains," *Contributions to Mechanics of Materials and Structures. CTU Reports*, Vol. 4, pp.

[14] SADE, http://klobouk.fsv.cvut.cz/~ondra/sade/sade.html

[15] F. Porkili. Integral histogram, "A fast way to extract histograms in cartesian spaces," *IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

김 향 화

2005년 7월 중국 연변대학교 전자정보공학부(공학사)
2008년 9월 중앙대학교 첨단영상대학원 영상공학과(공학석사)
관심분야 : 인공지능,컴퓨터게임



장 동 헌

2006년 2월 한국기술교육대학교 정보기술공학부(공학사)
2008년 9월 중앙대학교 첨단영상대학원 영상공학과(공학석사)
관심분야 : 컴퓨터 비전, 컴퓨터게임

김 태 용

1986년 2월 한양대학교 전기공학과(공학사)
1988년 2월 한양대학교 전자통신공학과(공학석사)
1998년 2월 포항공과대학교 컴퓨터공학과(공학박사)
1988년 3월~1999년 2월 한국통신 연구원
2003년 9월~현재 중앙대학교 첨단영상대학원 영상공학과 교수
관심분야 : 영상통신, 영상처리, 컴퓨터비전, 컴퓨터게임