「정보시스템연구」 제25권 제4호
한국정보시스템학회
2016년 12월, pp. 23~36

https://doi.org/10.5859/KAIS.2016.25.4.23

일반적인 비디오 게임의 AI 에이전트 생성을 위한 개선된 MCTS 알고리즘

오 평* · 김 지 민** · 김 선 정*** · 홍 석 민****

〈목 차〉

Ⅰ. 서 론

게임은 그래픽스, 사운드, 인터페이스, 스토리텔링 등 다양한 측면에서 이미 사용자의 기대치, 만족치를 상회할 만큼의 성장을 이미 이루었다. 게임 산업에서 이러한 요소들은 더 이상 사용자 경험을 증대시키기 위한 관심사가 아니다. 실제로도 많은 기업에서 타 게임과의 경쟁력을 갖추기 위해서 새로운 연구와 준비를 하고 있다. 바로 인공지능 분야가 그렇다. 인공지능에 관한 관심은 게임 업계뿐만 아니라 다른 IT 기업에서도, 심지어 국가 차원에서도 이에 대한 지원과 연구가 활발히 이루어지고 있다. 현재 많은 관심을 받고 있는 인공지능 연구 분야로는 유전 알고리즘, 강화 학습 알고리즘, 인공신경망 등으로 '다중 에이전트를 활용한 호텔 온라인 예약의 개념적 모형' (곽수환 · 강민철, 2000), '가상현실 : 3차원 가상 물리 시스템 개발' (임정환 등, 1997)과 같은 연구 사례가 있다.

게임에서 인공지능 분야에 대한 관심이 확대되고 많은 연구가 진행되고 있는 이유는 게임

* 한림대학교 인터랙션 디자인학 석사, vudhrh@gmail.com, 주저자
** 한림대학교 인터랙션 디자인학 석사, rudengkim@gmail.com
*** 한림대학교 융합소프트웨어학과 교수, sunkim@hallym.ac.kr
**** 한림대학교 광고홍보학과 교수, seokminhong@hallym.ac.kr, 교신저자

내에서 사용자와 상호작용을 하는데 많은 영향을 미치기 때문이다. 예를 들어 게임 사용자가 게임을 플레이함에 있어서 상호작용이 이루어지는 콘텐츠의 경우 과거에는 게임의 개발이 이루어지는 단계나 추후 업데이트 과정에서 정적으로 디자인되어 개발되었다. 그러나 이로 인한 콘텐츠는 매우 한정적이며, 사용자의 욕구를 충족시키기에는 턱없이 부족하기 때문에, 이를 자동적으로 생성하고 발전시키기 위한 연구가 이루어지면서 인공지능에 대한 관심이 커지기 시작했다.

게임에서 인공지능 에이전트를 개발하기 위해서는 각 게임의 장르, 특성, 인터페이스 등 다방면을 고려하고 게임과 관련된 전문 지식 등을 사용해 구현해야 한다. 그러나 이러한 에이전트는 특정 게임에만 적용 가능하며, 다른 게임에 적용하기 위해서는 또 다시 그 게임의 특성에 맞게 수정해야 한다는 번거로움이 있다. 이러한 문제를 해결하기 위해 일반적인 게임에서 범용적으로 적용 가능한 인공지능에 대한 연구가 활발히 진행되고 있으며, 이를 GGP (General Game Playing)라고 부른다(Lara-Cabrera et al., 2015).

하나의 소스 코드 작성으로 모든 게임에 적용 가능한 AI 에이전트를 생성한다는 목표는 매우 매력적이다. GGP 분야에 대한 연구는 아직 초기 단계여서 모든 게임을 아우르는 에이전트 생성보다는 비디오 게임이라는 다소 한정적인 분야에 초점을 맞춰 연구가 많이 진행되고 발전되고 있다. 본 논문에서는 이런 GGP 분야의 한 부분 GVGP(General Video Game Playing)의 인공지능 에이전트를 MCTS(Monte-Carlo Tree Search) 알고리즘을 개선하여 생성한다. 기존 인공지능 에이전트를 개발하기 위한 알고리즘은 다양한데, 대표적으로 FSM(Finite State Machine), HFSM(Hierarchical Finite State Machine), Behavior Tree 등이 있다. 그러나 이런 알고리즘들은 사전에 에이전트가 행동하게 될 로직 구조를 게임 관련 지식을 이용해서 미리 디자인해야 하기 때문에 GGP를 위한 AI 에이전트를 생성하기에는 다소 부적합하다. 게임에 대한 사전 트레이닝과 관련 지식 없이 AI 에이전트를 생성하기에 적합한 알고리즘으로, 본 논문에서는 MCTS를 제안한다. MCTS는 무작위 탐색을 이용해 목표한 값에 가장 근사한 값을 계산하기 위한 알고리즘으로, 현재 에이전트를 생성하기 위한 새로운 방법으로써 많은 연구가 진행되고 있는 알고리즘이며, 2006년 처음 창안된 이후로 바둑, 포커, 헥스 등 많은 장르에서 해당 알고리즘의 가능성을 입증했다(Lee et al., 2009; Rimmel et al., 2010; Lee et al., 2010).

최우선 깊이탐색 기법을 사용하는 MCTS 알고리즘은, 유한한 길이의 어떠한 게임에서도 적용 될 수 있으며(Chaslot et al., 2008), 게임에 대한 관련 지식을 필요로 하지 않는다. 이러한 특성들은 GGP분야에서 AI 에이전트를 구현하기에 적합한 특징이며, 구현적인 측면에서도 간단하다. MCTS는 기본적으로 완전 정보를 바탕으로 하는 알고리즘이다. 알고리즘 수행 단계 중 이루어지는 시뮬레이션 단계에서 제대로 된 시뮬레이션 값을 평가하기 위해서는 부득이하게 게임과 관련된 지식이 사용될 수밖에 없기 때문이다. 또한 MCTS에 대한 그 동안의 연구와 성능 검증들 또한 대부분 이런 완전 정보게임을 대상으로 이루어져 왔다. ‘완전 정보게임’

이란 게임에 대한 정보가 가려지지 않고 모두 드러나 있는 게임을 말한다. 예를 들면, 바둑은 나와 상대방의 유효한 정보(바둑돌)들이 모두 바둑판위에 드러나 있어, 해당 정보를 사용할 수 있으므로, 완전 정보게임이다. 반대로 '불완전 정보게임'이란 이와 같은 정보가 일부 혹은 전체가 가려져 있는 게임으로, 대표적인 예로는 상대방이 가진 카드를 볼 수 없는 포커가 있다. 그러나 본 논문에서 MCTS를 적용한 GVGP 분야는 일반적인 비디오 게임 모두를 대상으로 하는 것을 목표로 하기 때문에 하나의 컨트롤러가 이 많은 게임들의 관련 지식을 사용하는 것은 불가능하다. 즉 불완전 정보의 환경 안에서 구현되어 진다. 이와 관련된 연구 사례로는 포커나 스캣 게임에 MCTS 알고리즘을 적용해 가능성을 입증한 연구 결과가 있으나(Buro et al., 2009), 완전 정보 게임의 연구 사례에 비하면 아직 미비한 실정이다.

불완전 정보 게임에서 MCTS 알고리즘을 이용하여 AI 에이전트를 생성하기 위해, 본 논문은 기존 MCTS 알고리즘의 개선 방법을 제안한다. MCTS 알고리즘의 선택 단계에서 특정 결과가 반복되어 선택되지 않도록 미세한 패널티를 부과하여 효율적인 탐색이 가능하도록 개선하고, 확장 단계에서는 높은 보상값을 가질 가능성이 높은 자식 노드로 확장되도록 개선한다. 본 논문에서 제안하는 개선된 MCTS 알고리즘을 검증하기 위해, GVGP 연구가 활발히 진행되고 있는 커뮤니티인 GVG-AI Competition 사이트에서 제공되는 Framework를 사용하여 Video Arcade Game의 AI 에이전트를 생성하고, 테스트를 통해 기존 MCTS와 성능 차이를 비교한다.

Ⅱ. 관련 연구

2.1 General Game Playing

스탠포드에서 시작된 GGP는 AI 에이전트가 사전의 특정 트레이닝 없이 다른 게임에서 플레이될 수 있게 하기 위해 시작된 분야이다(Genesereth et al., 2005). 다시 말해 하나 이상의 게임에서 AI 프로그램이 잘 동작할 수 있도록 디자인하는 것을 의미한다(Lara-Cabrera et al., 2015). 게임에서 AI란 그 게임 안에서의 목적에 맞게 행동되어지도록 설계된다. 즉 체스를 위해 설계된 AI 에이전트는 바둑에서 작동하지 못한다. 그러나 GGP의 목적은 설계된 하나의 AI 에이전트가 체스에서도, 바둑에서도 그리고 그 이상의 다른 게임에서도 플레이될 수 있도록 하는 것이다. GGP 분야의 발전은 MCTS의 등장으로부터 많은 발전을 이루었다(Genesereth & Björnsson, 2013).

2.2 GVG-AI Competition & Framework

게임을 기초로 한 AI 경쟁 대회는 AI 알고리즘을 벤치마킹하기 위한 좋은 방법을 제공한다(Perez-Liebana et al., 2016). 그 중 하나인 GVG-AI Competition 사이트는 GVGP (General Video Game Playing)를 위한 AI 에이전트를 생성하기 위해 2014년 처음 시작 되었다(Frydenberg et al., 2015). 제작한 에이전트를 GVG-AI 사이트에 등록하면, 알려지지 않은 일련의 게임 셋을 가지고 이를 검증하며, 이 경쟁에 참여한 다른 개발자들과 비교해 순위를 매긴다. GVG-AI Competition 사이트에서는 개발

자들이 공통된 조건에서 에이전트를 생성할 수 있도록 Framework를 제공하며, 테스트를 위한 게임 셋으로 70여개의 아케이드 게임을 포함시켰다. 70개의 게임을 10개씩 묶은 총 7개의 Training Set이 존재한다.

개발자는 위 Competition 사이트에 Java 코드로 된 Controller를 제작하여 제출한다. 제출된 Controller는 위의 7개의 게임 셋 중 하나의 셋과 공개되지 않은 10개의 게임으로 이루어진 Validation Set을 통해 평가되고, 참여한 다른 개발자들의 Controller의 점수와 비교된 순위를 통해 점수를 얻는다. 7개의 Training Set은 모두 무작위로 선정되어 구성되었기 때문에, Set에 따라서 달라지는 특징은 없다. 따라서본 논문에서는 편의상 7개의 Training Set중 1번째 Training Set을 사용하였으며, 그 구성은 <표 1>과 같다.

2.3 Multi-Armed Bandit Problem

실제로 많은 상황에서 우리의 선택은 최상의 수치적 보상값을 얻을 수 있을 것이라 기대되는 선택을 하게 되지만 이러한 선택이 항상 최대화된 보상을 이끌어 내지는 못하는 경우가 보편적이다. 1952년 Robbins에 의해 처음 제안된 multi-armed bandit problem은 확률 이론에서, 도박꾼이 카지노에 존재하는 수많은 슬롯머신 중에서 가장 많은 보상을 얻기 위해서 어떤 슬롯머신의 슬롯을 몇 번 당겨야 가장 큰 보상을 얻을 수 있을까에 관한 문제이다. 이 문제에서 각 슬롯머신은 각자의 무작위 보상 값을 가지고 있을 수 있으며, 보상 값이 존재하지 않을 수도 있다. 도박꾼은 어떤 슬롯머신에서 어느 정도의 보상을 얻을 수 있는지에 대한 여부를 알 수 없다(Vermorel & Mohri, 2005). 이러한 경우에 대한 문제 해결 모델을 multi-armed bandit problem이라 통칭한다.

<표 1> Training Set (출처: http://gvgai.net/index.php)

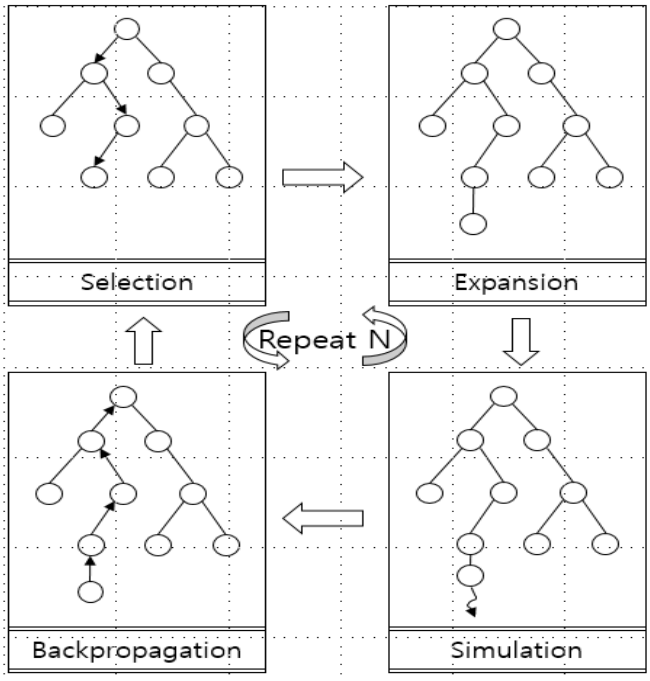
Training Set Games 1 (2015; CIG 2014)	
Aliens	이 게임은 화면 바닥에서 배를 조종하여 하늘에서 내려오는 외계인을 저지하는 게임으로, 외계인들이 바닥에 닿기 전에 그들을 모두 물리쳐야 승리한다.
Boulderdash	동굴을 배경으로 이루어지는 이 게임은, 동굴 안에 존재하는 다이아몬드를 곳곳에 숨어있는 적을 피해 찾아서 출구로 나가야 한다,
Butterflies	이 게임에는 나비와 그들의 고치가 있다. 나비들은 고치를 접하게되면 번식 하게 되고 모든 고치가 부화하기 전에 나비를 모두 사냥해야한다.
Chase	겁먹은 염소들을 찾아서 사냥해야 한다. 그러나 염소들은 죽은 동족을 보면 분노해 캐릭터를 공격하므로 조심해야 한다.
Frogs	흐르는 강물에 떠다니는 땟목에 올라타 강을 통과하고 출입구로 나가야 하는 게임이다.
Portals	맵에서 순간이동을 할 수 있는 문이 있고, 캐릭터를 조작해 알맞은 문을 찾아 미로를 빠져나가는 게임.
Sokoban	맵에서 캐릭터를 조작해 박스를 움직여 박스가 위치해야할 자리로 옮기는 게임
Survive Zombies	좀비를 피해 최대한 오래 살아 남는 게임
Missile Command	하늘에서 도시를 파괴하려는 미사일이 떨어지고 있다. 캐릭터를 조작하여 미사일이 도시로 도착하기 전에 모두 파괴해야 한다.
Zelda	적이 우글거리는 던전 속에서, 던전을 탈출하기 위한 열쇠를 찾아 던전을 탈출하는 게임

2.4 Monte-Carlo Tree Search

MCTS는 입력 변수 혹은 시스템의 확률적 특성으로 인한 결과를 정확하게 예측하기 어려운 확률 모델에서 사용되는 효과적인 방법으로, 수치적으로 일련의 난수를 반복적으로 발생시켜 시뮬레이션을 수행함으로써 원하는 해와 근사한 수치를 얻는 알고리즘이다(서경민 · 송해상, 2013). 초기의 MCTS는 1949년 Metropolis와 Ulam에 의해 제안되고 매우 광범위하게 설명되었으며, MCTS가 넓은 범위의 수학과 과학에 기반을 둔 문제에 대해 성공적으로 적용되어 사용될 수 있음을 증명하였다. 그리고 또한 게임 분야에서도 솔리테르 카드 게임에서 성공적인 보상의 가능성을 계산하여 해당 알고리즘의 가능성을 입증하였다. 이후 2006년 MCTS는 보드게임과 비디오 게임을 위한 성공적인 알고리즘으로 설계되었으며, 아케이드 게임을 넘어 현재의 3D 실시간 전략 게임에서도 성공적인 적용 사례를 보였다. MCTS가 게임에 적용됨에 있어 가지는 3가지 장점은 다음과 같다(Ross, 2014).

비디오 게임의 불확실성에 대응하기 적합하다.
다른 선택지에 대한 반복적 시도로 인해 최악의 선택을 할 가능성이 현저히 낮다.
원하는 어느 시점에서든지 알고리즘의 수행을 멈추고 결과값을 반환할 수 있기 때문에 계산적인 측면에서 매우 효율적이다.

또한, 이러한 MCTS를 구현하여 적용하기 위해서는 다음과 같은 세 가지 조건이 만족되어야 하는데, 그 조건은 다음과 같다(Lee, 2014; Browne, et al., 2012).



<그림 1> MCTS 알고리즘

[조건 1] 시뮬레이션을 통한 득실 값의 한계가 있어야 한다. (최대 / 최솟값이 있어야 한다.)

[조건 2] 게임의 규칙이 정해져 있어야 하며, 완전 정보 게임이어야 한다.

[조건 3] 게임의 길이가 제한되어 비교적 빠른 시뮬레이션이 수행되어야 한다.

MCTS의 알고리즘은 Selection, Expansion, Simulation, Backpropagation의 4단계를 <그림 1>과 같이 반복하며 수행된다(Jacobsen et al., 2014).

[단계 1] Selection : 시작 노드에서 시작해 단말 노드를 만날 때까지 재귀적으로 호출되며 수행된다.

[단계 2] Expansion : 단말 노드를 만난 후 해당 노드가 게임의 마지막 상태가 아닐 경우, 하나 혹은 그 이상의 노드를 자식 노드로 생성한다.

[단계 3] Simulation : 단말 노드 중 하나의 노드가 선택되었을 때, 가상으로 플레이되며 이는 보통 Roll-Out이라고 부른다. 이 시점에서 수행되어지는 행동은 게임의 마지막 단계가 될 때까지 임의로 선택되어 수행된다.

[단계 4] Backpropagation : 시뮬레이션이 수행된 이후 해당 노드는 결과를 반영해 가며 시작 노드를 만날 때까지 수행된다.

본 논문에서 MCTS 알고리즘은 Selection 단계에서 선택된 현재 게임의 상태 노드가 할 수 있는 행동 중에서 하나를 무작위로 입력받아 자식 노드로 추가하고(Expansion 단계) 그 행동을 GVG-AI Framework에 내장된 평가 함수를 통해 점수를 반환 받은 후(Simulation 단계), 루트노드로 되돌아가며 반환 받은 값을 누적 시켜 나간다.

2.4.1 Upper Confidence Bound(UCB)

Upper Confidence Bound Method는 1985년 Lai와 Robbins에 의해 제안되고 1995년 Agrawal에 의해 소개되고 분석된 알고리즘이다(Garivier & Moulines, 2008). UCB 공식은 각 레벨에서 가장 최선의 자식 노드를 선택한다. 이를 MCTS에 적용하기 위해서 UCB 공식을 MCTS의 탐색 트리에 적용하며 이를 UCT(Upper Confidence Bound for Tree)라고 부르고 <식 1>과 같다.

$$\overline{X_j} + 2C_p\sqrt{\frac{2\ln n}{n_j}} \qquad \text{<식 1>}$$

$\overline{X_j}$는 자식 노드 j의 평균 스코어를 의미하고, n은 해당 노드의 부모 노드의 방문 횟수, n_j는 해당 자식 노드의 방문 횟수를 의미한다. C_p는 Exploration Term에 적용될 상수를 의미하며, 일반적으로는 $\sqrt{2}$를 사용한다. MCTS 알고리즘에서 UCT 공식은 무작위로 확장된 탐색 공간에서 가장 최선이라고 짐작되는 노드를 선택한다.

2.4.2 Exploration & Exploitation

UCT 공식은 exploitation term과 exploration term으로 총 2개의 term으로 구성되어 있다. 첫 번째 term은 $\overline{X_j}$이며, 이는 Backpropagation 단계에서 루트 노드로 되돌아가며 증가하는 누적

된 보상값을 의미한다. 두 번째 term은 $\sqrt{\frac{2\ln n}{n_j}}$ 을 의미하고, 각 알고리즘 수행단계에서 부모 노드가 방문 될 때 증가하는 값이다.

2.4.3 Anytime Algorithm

MCTS의 가장 큰 장점 중 하나는 알고리즘의 수행 중간에 언제든지 수행을 멈추고 그 순간까지의 결과에 대한 최대화된 보상값을 얻어낼 수 있다는 것이다(Frydenberg et al., 2015; Garivier & Moulines, 2008). 한 번의 tick에서 40ms의 시간제한이 주어지는 GVG-AI Framework에서 이러한 특징은 매우 강력한 장점이다.

Ⅲ. Enhanced MCTS

기존의 MCTS를 개선하기 위해 다음 네 가지 기술을 적용한다.

3.1 MixMax Backups

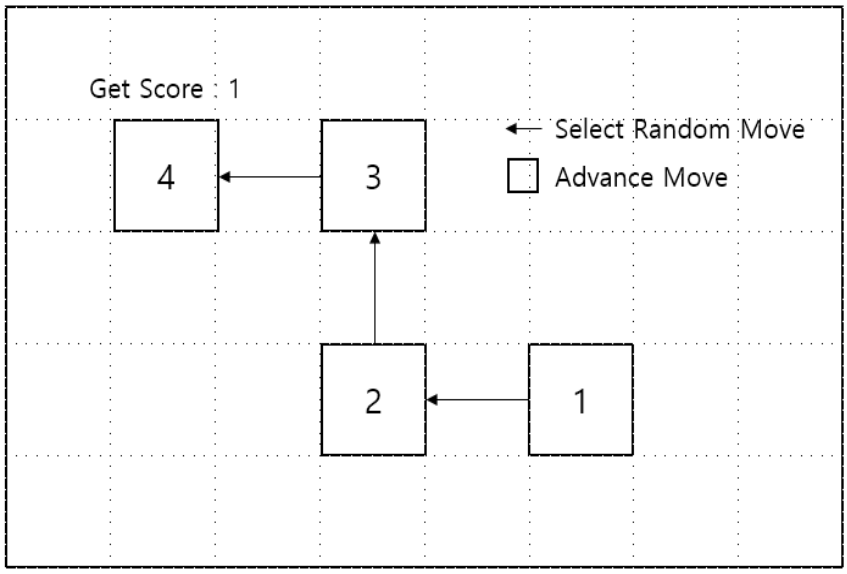
MixMax Backups는 위험을 탐색하는 수치를 조절하는 방법으로써, UCT 공식의 exploitation term에 적용하여 사용하며 공식은 <식 2>와 같다(Jacobsen et al., 2014; Frydenberg et al., 2015; Garivier & Moulines, 2008).

$$Q \cdot Maxscore + (1-Q) \cdot \overline{X_j} \qquad \text{<식 2>}$$

Q는 [0, 1] 사이의 범위 값을 가지며, 게임의 maxscore에 곱해지게 된다. 즉 0일 경우 얻어지는 스코어가 없으므로, 유의미한 값을 가지는 탐색 트리의 확장이 어렵고 이는 실제 게임플레이에서 캐릭터가 겁을 먹은 것과 같은 안전한 루트의 자식 노드를 선택하게 하며, 1에 가까워질수록 결과에 대한 위험성 보다는 큰 보상 값에 치우친 탐색을 하게 된다. 따라서 너무 높은 수치의 Q 값의 설정은 게임 내 몬스터에게 플레이어 캐릭터가 사망하는 등의 결과를 초래 할 수 있다 본 논문에서는 경험에 의해 해당 Q 값을 0.25로 설정하였다. MixMax Backups는 UCT알고리즘에서 탐색과 보상값에 대한 Trade-Off를 조정하는 목적으로 사용된다. 알고리즘이 보상값에 치우친 수행을 하게 되면, 올바른 탐색을 할 수 없고 탐색에 치우친 수행을 하게 되면, 적절한 보상값을 얻지 못한다. MixMax Backups는 Q값 설정으로 이 양자관계의 균형을 조정한다.

3.2 Expansion

기존 MCTS 알고리즘에서의 Expansion은 행동 가능한 선택 중에서 무작위로 하나의 자식 노드를 확장시키는 간단한 구조로 이루어져 있다. 하지만 본 논문에서는 선택 가능한 행동 중에서 높은 점수를 얻을 수 있는 선택지가 있는지 먼저 확인한 후 있을 경우에 그 행동을 자식 노드로 추가하고, 없을 경우 무작위로 자식 노드를 추가하도록 하였다.



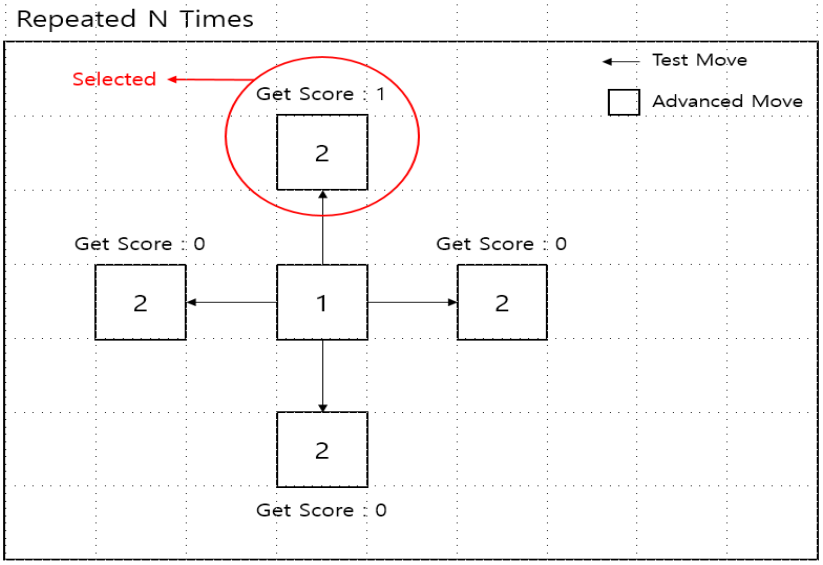
<그림 2> 기본 Roll-Out

3.3 Modified Reversal Penalty

기존의 MCTS Controller는 게임 내에서 몇몇 인접한 위치로만 나아가거나 원래 있던 위치로 다시 되돌아오는 등, 진동의 형태와 유사한 움직임을 종종 보인다(Frydenberg et al., 2015). UCT 공식을 적용할 때 exploitation에 영향을 주는 game score가 몇몇의 큰 점수에만 의존하는 현상 때문에, 무의미한 탐색이 이루어지는 경우가 많기 때문이다. Frydenberg et al.(2015)는 이러한 문제점을 해결하기 위해서 Reversal Penalty 방법을 제안했으며, 기존의 MCTS와 비교했을 때, 확실히 개선된 성능을 보였다. 이 방법은 최근 에이전트가 수행한 5가지의 행동을 저장해 놓고, 다음 행동을 선택할 때 UCT 값에 의해 선택된 행동이 저장해놓은 행동과 같은 행동일 경우 미세한 페널티 값을 부여하는 방법으로, 해당 논문에서는 0.95를 곱하는 방식으로 사용 하였다. Frydenberg et al.(2015)에서 0.95 라는 적은 값의 페널티를 주는 이유는 저장된 행동과 일치하지 않은 다른 일반적인 행동에 대해서는 큰 영향을 주지 않기 위함인데, 만약 행동 가능한 행동의 수가 3개인 게임이거나 다소 적을 경우, 같은 행동이 여러 번 페널티를 받되, 의도한 값보다 많은 페널티가 주어질 수 있다. 따라서 본 논문에서는 최근 5개의 행동을 저장하는 방식이 아닌 Controller의 선택 가능 행동 수에 따라 가변적으로 저장 개수를 변경하여 의도한 페널티보다 많은 페널티가 부여되어 행동 선택에서 제외되지 않도록 한다.

3.4 Roll-Out

GVG-AI Framework에 구현된 기본 MCTS의 roll-out은 무작위로 선택된 행동을 시뮬레이



<그림 3> 수정된 Roll-Out

션을 통해 트리의 특정 깊이까지 진행한 후, 그 시점의 플레이어의 선택된 행동을 기준으로 산정된 게임 스코어를 반환하는 방식이다. 이 방식은 무작위로 선택된 행동만을 시뮬레이션하기 때문에 선택되지 않은 행동에 대해서는 고려하지 못해 정확한 시뮬레이션이 불가능하다 (<그림 2>).

그러나 수정한 roll-out에서는 현재 위치에서 선택 가능한 모든 행동들에 대해서 시뮬레이션을 우선 수행해 스코어를 계산하고 가장 높은 스코어를 얻은 행동을 선택해 적용하며, 이 과정을 탐색 트리의 특정 깊이까지 반복한다. 가능한 모든 행동에 대한 시뮬레이션을 통해서 기존보다 좀 더 정확한 시뮬레이션이 가능하다 (<그림 3>).

Ⅳ. 실험 결과

본 논문에서는 제안 알고리즘을 Training Set 1과 Validation Set에서 테스트하였다. Training Set 1은 게임에 대한 정보가 제공된 10개의 게임으로 구성되어 있고, Validation Set은 게임에 대한 정보가 제공되지 않은 10개 게임으로 구성되어 있다. Training Set 1과 Validation Set 각각 10개의 게임을 5번씩 [MixMax], [MixMax, Reversal Penalty], [MixMax, Reversal Penalty], [MixMax, Modified Reversal Penalty], [MixMax, Modified Reversal Penalty, Roll-Out], [MixMax, Expansion, Modified Reversal Penalty, Roll-Out], 총 5개의 조합의 Controller에서 테스트 하였다.

<표 2> Training Set1의 실험 결과

Controller	G1		G2		G3		G4		G5		G6		G7		G8		G9		G10	
	P	W	P	W	P	W	P	W	P	W	P	W	P	W	P	W	P	W	P	W
[Mixmax]	63.2	1	14.8	0.2	30.4	0.8	1.4	0	0.4	0.4	2.6	0.4	0.2	0.2	1.4	0.2	-4.4	0.2	4.6	0
[Mixmax, Reversal Penalty]	55.8	0.4	14	0.2	28.8	1	2.8	0	0.2	0.2	0.6	0.4	0	0	1	0	-5	0.2	3.6	0.8
[Mixmax, Modified Reversal Penalty]	53	0	14.4	0.4	29.6	1	3.4	0.2	0.2	0.2	2.2	0.4	0.2	0.2	1.4	0.2	-4	0.2	3	0.6
[Mixmax, Modified Reversal Penalty, Roll-Out]	48.6	0.2	7	0.2	32.4	0.8	2.6	0	0.2	0.2	3.4	0.8	0.4	0.4	1	0	-6.6	0.2	4.6	0.8
[Mixmax, Modified Expansion, Modified Reversal Penalty, Roll-Out]	52.6	0.8	10.2	0.2	28	0.8	3.6	0.2	0.6	0.6	3.4	0.4	0.2	0.2	1.2	0.2	-7	0	6.4	0.6

<표 3> Validation Set의 실험 결과

Controller	G1		G2		G3		G4		G5		G6		G7		G8		G9		G10	
	P	W	P	W	P	W	P	W	P	W	P	W	P	W	P	W	P	W	P	W
[Mixmax]	44.4	0	46.4	1	5	0	0.6	0	0.2	0	1	1	2.8	1	2.8	0	0	0	10.2	0
[Mixmax, Reversal Penalty]	65.8	0	27.2	0.8	6.4	0	0.8	0	0	0	0.8	0.8	6.8	1	0	0	0.2	0.2	11.8	0.8
[Mixmax, Modified Reversal Penalty]	63.6	0	38	1	8.2	0	0.8	0	4.6	0	1	1	8.4	1	0	0	0.2	0.2	13.4	0.8
[Mixmax, Modified Reversal Penalty, Roll-Out]	110	0.2	35.2	1	12.6	0	1.2	0	2.4	0	0.8	0.8	18.2	1	3.2	0	0	0	13	0.4
[Mixmax, Modified Expansion, Modified Reversal Penalty, Roll-Out]	26.8	0	60.4	1	29.6	0	0.8	0	8.2	0.2	0.6	0.6	34	1	7.2	0	0.2	0.2	12.8	0.4

실험 결과 <표 2>, <표 3>과 같은 결과를 얻었다. 표에 보이는 'P'의 의미는 게임이 끝난 시점에서 얻은 포인트를 의미하며, 'W'는 5번의 실험동안 게임을 클리어 한 승률을 의미한다 (최대 1). G_i는 실험 Set에 포함된 각각의 게임이다. 표의 각 데이터 셀에서 보이는 굵고 붉게 표시된 데이터는 다른 비교 대상 알고리즘에 비해 우수한 결과인 것을 표시한 것이다. 먼저 Training Set 1의 경우엔 5개의 실험 Controller 모두 큰 차이 없이 고르게 점수가 분포되었다. 그러나 실제로 GVG-AI Competition의 목적에 부합하는 Set인, Validation Set의 테스트에서는 최선의 탐색이 가능하도록 UCT 알고리즘을 개선한 MixMax Backups와 효율적인 노드 확장을 위한 Modified Expansion, 중복된 탐색을 반복하지 않도록 패널티를 부여하는 Modified

Reversal Penalty, 그리고 정확한 시뮬레이션이 이루어지도록 개선한 Roll-Out을 모두 적용한 [MixMax, Expansion, Modified Reversal Penalty, Roll-Out] (M/E/MP/R) Controller가 가장 우수한 성능을 보였다. 위 Controller의 다른 4개의 Controller에 비해서도 우수하였으며, 또한 GVG-AI Competition 시뮬레이션 값에서도 78 Point로 가장 우수한 성적을 거두었으며, 2014년 Competition에서 가장 우수한 성적을 보였던 “adrienctx”의 Controller보다 6점 높은 점수를 얻었다.

Ⅴ. 연구결과 및 향후 연구과제

본 논문에서는 GVGP 장르에서 사전 트레이닝이나 게임 지식을 활용하지 않는 범용적인 목적으로써 활용될 수 있는 AI 에이전트를 생성하기 위해 개선된 MCTS 알고리즘을 제안하였다. 실험 결과 GVG-AI Competition 기준으로 일반적인 MCTS 알고리즘에 비해 성능 향상을 확인할 수 있었다. MCTS 알고리즘은 GGP 장르에서 AI 에이전트를 생성하기에 많은 부분이 적합하고 타 알고리즘과 비교했을 때 좋은 성능을 보인다. 비록 현재는 아케이드 게임에 한정이 되어 적용되었지만, 본 논문에서 개선한 MCTS 알고리즘은 게임에 대한 지식 없이도 게임을 스스로 진행시키는 에이전트 생성의 성공적인 결과를 보였다. 이로써 하나의 소스코드로 모든 게임의 에이전트에 적용한다는 궁극적인 목표에 기여했다고 볼 수 있다. MCTS 알고리즘을 GGP의 목적성에 맞게 개선하고 보완한다면, 해당 목표를 달성하는 알고리즘으로써 MCTS 가 그 역할을 할 수 있을 것 이라 기대한다.

향후 연구로는 AI 에이전트가 움직일 경로와 그에 대한 보상값을 계산해 내는 연산이 필요한데, 실시간으로 빠르게 계산해 내야하는 게임 환경에서는 시간적 제약으로 충분한 보상값을 찾지 못하는 경우가 종종 발생하기 때문이다. 이 제약적 환경에서 확실한 보상값을 얻기 위해, 가상의 환경 변수를 제안하여 보상식을 개선할 예정이다.

참고문헌

곽수환, 강민철, “인공지능/다중 에이전트를 활용한 호텔 온라인 예약의 개념적 모형”, 한국정보시스템학회, 단일호, pp.107-113, 2000

임정환, 김태현, 김현수, 이재기, 최형림. “인공지능/가상현실: 3차원 가상 물리 실험 시스템 개발”, 한국정보시스템학회, 단일호, pp75-82, 1997.

서경민, 송해상. “효율적인 몬테 칼로 시뮬레이션을 위한 중요 샘플링 기법이 내장된 실험 틀 설계,” 한국콘텐츠학회논문지, Vol. 13, No. 4, pp53-63, 2013.

Browne, C., Lucas, S., Tavener, S., Perez, D., Samothrakis, S., and Colton, S., “A Survey of Monte Carlo Tree Search Methods,” *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 4, No. 1, pp. 1-49, 2012.

Buro, M., Long, J. R., Furtak, T. and Sturtevant, N., “Improving State Evaluation,

Inference, and Search in Trick-Based Card Games," In Proceedings of the 21st International Joint Conference Artificial Intelligence (IJCAI), 2009.

Chaslot, G., Bakkes, S., Szita, I. and Spronck, P., "Monte-Carlo Tree Search: A New Framework for Game AI," Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE), 2008.

Frydenberg, F., Andersen, K. R., Risi, S. and Togelius, J., "Investigating MCTS Modifications in General Video Game Playing.", 2015 IEEE Conference on Computational Intelligence and Games (CIG). 2015.

Garivier, A, and Moulines, E., "On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems," arXiv preprint arXiv:0805.3415, 2008.

Genesereth, M., Love, N. and Pell, B., "General Game Playing: Overview of the AAAI Competition," *AI Magazine*, Vol. 26, No. 2, pp. 62-72, 2005.

Genesereth, M. and Björnsson, Y., "The International General Game Playing Competition," *AI Magazine*, Vol. 34, No. 2, pp. 107-111, 2013.

Jacobsen, E. J., Greve, R. and Togelius, J., "Monte Mario: Platforming with MCTS," Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 293-300, 2014.

Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C. and Fernández-Leiva, A. J., "Game Artificial Intelligence: Challenges for the Scientific Community," In Proceedings 2st Congreso de la sociedad Española para las Ciencias del Videojuego Barcelona, Spain, 2015.

Lee, B.-D., "Monte-Carlo Tree Search Applied to the Game of Tic-Tac-Toe," *Journal of Korea Game Society*, Vol. 14, No. 3, pp. 47-54, 2014.

Lee, C.-S., Wang, M.-H., Chaslot, G., Hock, J.-B., Rimmel, A., Teytaud, O., Tsai, S.-R., Hsu, S.-C. and Hong, T.-P., "The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 1, No. 1, pp. 73-89, 2009.

Lee, C.-S., Müller, M. and Teytaud, O., "Special Issue on Monte Carlo Techniques and Computer Go," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, No. 4, pp. 225-228, 2010.

Perez-Liebana, D., Samothrakis, S., Togelius, J., Schaul, T., Lucas, S. M., Couëtoux, A., Lee, J., Lim, C.-C. and Thompson, T., "The 2014 General Video Game Playing Competition," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 8, No. 3, pp. 229-243, 2016

Rimmel, A., Teytaud, O., Lee, C.-S., Yen, S.-J., Wang, M.-H. and Tsai, S.-R., "Current Frontiers in Computer Go," *IEEE*

Transactions on Computational Intelligence and AI in Games, Vol. 2, No. 4, pp. 229-238, 2010.

Ross, B., "General Video Game Playing with Goal Orientation.", Diss. Master's Thesis, University of Strathclyde, 2014.

Vermorel, J. and Mohri, M., "Multi-armed Bandit Algorithms and Empirical Evaluation," *European Conference on Machine Learning*, Vol. 3720, pp. 437-448, 2005.

오 평 (Pyeong Oh)

현재 한림대학교 인터랙션 디자인 대학원 인터랙션 디자인학 석사 과정 중에 있다. 한림대학교 유비쿼터스 게임공학과 학사 학위를 취득하였으며(2014), 현재 관심분야는 게임 인공지능, 알고리즘, 딥 러닝 등이다.

김 지 민 (Ji-Min Kim)

한림대학교 인터랙션 디자인 대학원 인터랙션 디자인학 석사 과정 재학중이며 한림대학교 유비쿼터스 게임공학과 학사 학위를 취득하였다(2014). 현재 관심분야는 게임 인공지능, 딥 러닝, 절차적 콘텐츠 생성 등이다.

김 선 정 (Sun-Jeong Kim)

현재 한림대학교 융합소프트웨어학과 교수로 재직 중에 있다. 고려대학교에서 컴퓨터학 석사(1998)와 박사(2003) 학위를 취득하였다. 관심분야는 컴퓨터 그래픽스, 게임 엔진, 가상/증강현실 등이다.

홍 석 민 (Hong, Seokmin)

현재 한림대학교 광고홍보학과 교수로 재직 중에 있다. 미국 University of Texas at Austin에서 광고학 석사(1998)와 박사(2003)학위를 취득하였다. 관심분야로는 마케팅, 뉴미디어, 그리고 사회연결망이다.

<Abstract>

Enhanced MCTS Algorithm for Generating AI Agents in General Video Games

Pyeong Oh · Ji-Min Kim · Sun-Jeong Kim · Seokmin Hong

Purpose

Recently, many researchers have paid much attention to the Artificial Intelligence fields of GVGP, PCG. The paper suggests that the improved MCTS algorithm to apply for the framework can generate better AI agent.

Design/methodology/approach

As noted, the MCTS generate magnificent performance without an advanced training and in turn, fit applying to the field of GVGP which does not need prior knowledge. The improved and modified MCTS shows that the survival rate is increased interestingly and the search can be done in a significant way. The study was done with 2 different sets.

Findings

The results showed that the 10 training set which was not given any prior knowledge and the other training set which played a role as validation set generated better performance than the existed MCTS algorithm. Besed upon the results, the further study was suggested.

Keywords: Monte-Carlo Tree Searching, General Game Playing, GVG-AI (General Video Games Artificial Intelligence)

* 이 논문은 2016년 10월 4일 접수, 2016년 10월 28일 1차 심사, 2016년 12월 13일 게재 확정되었습니다.