ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 21, No. 8, pp. 572-577, 2015. 8
http://dx.doi.org/10.5626/KTCP.2015.21.8.572

일반 비디오 게임 플레이 인공지능을 위한 GreedyUCB1기반 몬테카를로 트리 탐색

(GreedyUCB1 based Monte-Carlo Tree Search for General Video Game Playing Artificial Intelligence)

박 현 수 [†] 김 현 태 [††] 김 경 중 [†††]
(Hyunsoo Park) (HyunTae Kim) (KyungJoong Kim)

요 약 보통의 인공지능 시스템은 특정 작업을 수행하기 위해 설계되며, 해당 작업만을 수행하는 능력을 가진다. 그에 반해 인공 일반지능이란 설계 당시 목표로 한 작업만이 아니라 새로 접하는 다양한 문제에도 대응할 수 있는 인공지능을 의미한다. 최근 게임 인공지능 분야의 일반지능 문제인 General Video Game Playing에 대한 관심이 높아지고 있다. 비디오 게임으로 범위가 제한되었지만, 다양한 형태의 비디오 게임을 플레이 할 수 있는 단일 인공지능을 설계하는 것은 상당히 도전적인 문제이다. 본 논문에서는 Monte-Carlo Tree Search를 이용하는 기존 비디오 게임을 위한 인공 일반지능을 개선하는 방법에 대해 기술한다. 여기서는 UCB1 알고리즘을 문제에 적합하도록 개선한 GreedyUCB1과 게임 분석을 통해 얻은 지식을 활용한 Rollout 방법을 제안한다. 제안한 방법으로 개발된 인공지능은 국제 학술대회인 IEEE Computational Intelligence in Games의 2014년 인공지능 경진 대회에 출전하여 4위의 성적을 보였다.

키워드: General Video Game Playing, 게임 인공지능, Monte-Carlo Tree Search, UCB1, 인공지능 경진대회

Abstract Generally, the existing Artificial Intelligence (AI) systems were designed for specific purposes and their capabilities handle only specific problems. Alternatively, Artificial General Intelligence can solve new problems as well as those that are already known. Recently, General Video Game Playing the game AI version of General Artificial Intelligence, has garnered a large amount of interest among Game Artificial Intelligence communities. Although video games are the sole concern, the design of a single AI that is capable of playing various video games is not an easy process. In this paper, we propose a GreedyUCB1 algorithm and rollout method that were formulated using the knowledge from a game analysis for the Monte-Carlo Tree Search game AI. An AI that used our method was ranked fourth at the GVG-AI (General Video Game-Artificial Intelligence) competition of the IEEE international conference of CIG (Computational Intelligence in Games) 2014.

Keywords: general video game playing, general artificial intelligence, monte-carlo tree search, UCB1, AI competition

· 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 중견연구자지원사업(2013-R1A2A2A01016589)임
· 이 논문은 제41회 동계학술발표회에서 '게임 분석과 MCTS를 이용한 비디오 게임 일반 인공지능'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 세종대학교 컴퓨터공학과
rex8312@gmail.com
†† 비 회 원 : 와이즈넛
kimhntae@gmail.com
††† 정 회 원 : 세종대학교 컴퓨터공학과 교수(Sejong Univ.)
kimkj@sejong.ac.kr
(Corresponding author임)

논문접수 : 2015년 3월 31일
(Received 31 May 2015)
논문수정 : 2015년 6월 1일
(Revised 1 June 2015)
심사완료 : 2015년 6월 8일
(Accepted 8 June 2015)

정보과학회 컴퓨팅의 실제 논문지 제21권 제8호(2015. 8)

1. 서 론

인공지능 연구의 초창기부터, 사람과 같은 일반지능(General Intelligence)을 구현하려는 것은 모든 인공지능 연구자들의 궁극적인 목표중의 하나였을 것이다. 최근 게임 인공지능 분야의 인공 일반지능(Artificial General Intelligence, AGI) 문제인 일반 게임 플레이(General Game Playing, GGP) 인공지능에 대한 연구를 활발히 진행하고 있다[1][2]. GGP의 목적은, 다양한 게임을 플레이 할 수 있는 게임 인공지능을 설계하는 것이다. 이것은 매우 도전적이고 어려운 주제이지만, 게임 인공지능의 다양한 분야에서 활용을 기대할 수 있다.

초기의 GGP 연구는 보드게임을 다루었으나, 최근 이 개념을 비디오 게임으로 확장한 GVGP (General Video Game Playing) 또한 활발히 연구되고 있다[2]. 최근 국제 학술대회인 2014년 IEEE CIG (Computational Intelligence in Games)에서 GVG-AI (General Video Game-Artificial Intelligence)플랫폼을 이용한 GVGP 인공지능 구현 경진대회가 개최되었다[3]. 2014년 대회의 결과, MCTS (Monte-Carlo Tree Search)를 기반으로 하거나, 유사한 탐색기법을 사용한 인공지능들이 높은 성적을 보였다. 상위 다섯 개의 인공지능 중에 셋이 MCTS를 사용했다[3].

최근, MCTS는 뛰어난 성능 때문에 GVGP 분야뿐 아니라 게임 인공지능 분야 전반에서 폭넓게 활용되고 있다[4]. MCTS는 현재 게임 상태 s_0를 미래에 더 나은 상태로 바꿀 수 있는 행동 a_0 을 찾는 것이 목적인데, 게임 상태를 평가하기 위해 게임에 관한 지식을 이용 하는 대신에 플랫폼에서 제공하는 시뮬레이터를 이용하기 때문에, 특정 게임에 종속되는 지식을 사용할 수 없는 GVGP에 적합하다. 또한 주어진 시간에 맞추어 실행시간을 조절하기 쉽기 때문에, 실시간으로 진행되는 비디오 게임에 적합하다. 인공지능은 게임 틱(tick, 게임 내 시간의 최소단위, 보통 16ms부터 40ms정도)안에 현재 상황에 적합한 행동을 하는데, MCTS는 언제나 알고리즘을 중단하고, 현재까지 최선의 탐색 결과를 이용할 수 있다.

본 논문은 GVG-AI 플랫폼을 활용하여, 기존 인공지능을 개선하기 위한 방법을 제안한다. 본 논문에서는 MCTS기반 인공지능의 Select을 개선하기 위해 UCB1 [5]을 변형한 GreedyUCB1을 제안하였고, Rollout에 게임에 대한 지식을 반영하여 행동을 선택할 확률을 수정하였다. MCTS는 기존 탐색 알고리즘에 비해 효율적이지만, 보상이 드문 게임의 경우 깊은 탐색이 힘든 단점이 있다. 이때 GreedyUCB1는 작은 보상을 강화시켜 보상이 있는 상태로 탐색이 쉽게 만들어 준다. 그리고 무기를 사용하는 게임들이 유사한 게임 방식을 가진다는 점에 착안한 Rollout 방법은 특정 게임들에서 성능을 높여준다. 본 논문에서 제안하는 방법으로 구현된 인공지능은 IEEE CIG 2014의 GVG-AI 경진대회에 참가하여, 4위를 기록하였다[3].

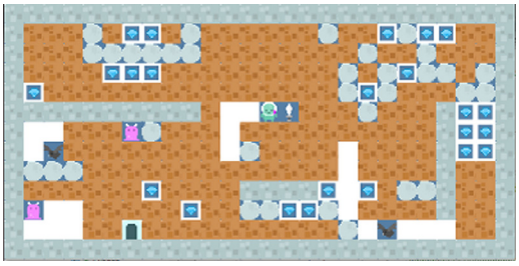
2. 관련 연구

General Game Playing (GGP)는 여러 가지 게임을 하나의 인공지능으로 플레이하는 능력을 의미한다. 2005년, AAAI (Association for the Advancement of Artificial Intelligence)의 국제 학술 대회에서 개최된 경진대회 이후 GGP가 주목받기 시작했다[6]. 이 경진대회의 목표는 다양한 보드 게임을 플레이 할 수 있는 인공지능을 구현하는 것이다. 이때 공개된 GDL (Game Description Language)[1]과 플랫폼을 기반으로 GGP연구가 활성화 되었다. 플랫폼은 GGP로 정의된 게임 규칙을 실행하는 일종의 게임엔진이다.

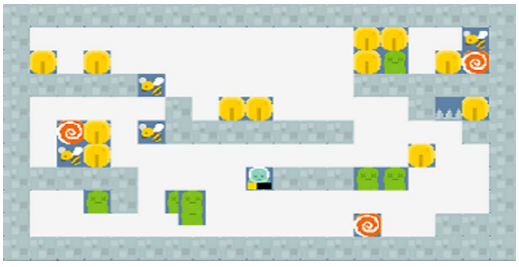
GGP의 개념을 비디오 게임으로 확장한 GVGP (General Video Game Playing)를 위한 첫 번째 시도는 2008년 Atari 2600 에뮬레이터를 이용한 플랫폼인 ALE (Arcade Learning Environment)의 개발과 함께 시작되었다[7]. 그 이후, ALE에 강화학습, 인공 신경망 등을 이용한 다양한 연구가 진행되었다[8-10]. ALE는 다양한 장점이 있었으나, 게임 상태를 알기위해서는 영상처리를 하거나, 메모리를 직접 읽어야 하는 번거로움이 있었다.

때문에, D. Perez et al.은 새로운 GVGP 플랫폼인 GVG-AI 플랫폼을 개발하여 공개했다[3]. 그리고 GGP의 GDL처럼 새로운 프로그래밍 언어인 VGDL (Video Game Description Language)를 이용해 게임을 정의했다[11]. VGDL과 GVG-AI 플랫폼의 관계는 GDL과 GGP 플랫폼의 관계와 유사하다. GDL을 이용해 보드게임을 정의하는 것처럼 VGDL을 이용하면 Pac-Man과 같은 3인칭 시점, 2D Grid 환경을 배경인 비디오 게임을 정의할 수 있다.

2014년 국제 학술대회 IEEE CIG (Computational Intelligence in Games)의 GVG-AI 경진대회에서는 훈련용(training), 검증용(validation) 그리고 평가용(test) 각각 10개씩 총 30개의 게임을 이용하였다. 이것은 패턴 분류분야의 교차 검증(cross validation)을 모방한 것이다. 사전에 참가자들에게는 학습용 세트 게임만 인공지능 개발 목적으로 제공된다. 검증용 게임 셋은 숨겨진 상태로 인공지능을 테스트하기 위한 용도로 이용할 수 있다, 그리고 대회는 공개된 적이 없는 테스트 셋으로 진행된다. 패턴 분류의 과적합(overfitting)문제와 마찬가지로 GVGP에서도 특정 게임들에 최적의 성능을 내도록 설계된 인공지능은 해당 게임들에는 좋은 성능을



(a) Boulder Dash



(b) Survive Zombies

그림 1 VGDL로 만든 GVG-AI 게임 예시
Fig. 1 Example games defined by VGDL

보일 수 있지만, 다른 게임들에서 낮은 성능을 보일 가능성이 높다. 그림 1은 당시 공개된 훈련용 게임들 중 일부이다.

2014년 S. Lucas는 진화 연산과 MCTS를 결합한 Fast-Evo MCTS 알고리즘을 선보였다[12]. 가벼운 진화 연산을 사용하여 현재 주어진 게임 환경에 적응할 수 있는 MCTS를 설계하였다. D. Perz는 Fast-Evo 연구를 확장하여 GVG-AI 플랫폼에 적용하고, 추가로 시뮬레이션을 통한 경험을 저장하고, 이를 탐색에 활용하는 방법을 제안하였다[13]. MCTS가 효율적이긴 하지만, 이 플랫폼에서 제공하는 많은 게임에서 불충분한 탐색 성능을 보여주었다. 적절한 보상을 주는 행동을 찾지 못하기 때문인데, Prez는 다른 객체들과 충돌했던 경험과 충돌하지 않은 객체에 대한 호기심을 이용해서 탐색의 효율성을 높이는 방법을 선보였다.

3. 제안하는 방법

게임 인공지능이 해결해야 하는 문제를 단순화 하면, 주어진 게임 상태 s_0에서 가능한 선택들 $a_0, a_1, \dots a_n$ 중에 가장 높은 기대 보상(게임 점수, 승리와 관련된 값)을 가진 선택 a^*를 찾아내는 것이다. 보통 게임 인공지능을 설계할 때, 개발자가 가진 지식(어떤 상황에서 어떤 행동이 가장 적합한지)을 모델링하고, 게임 인공지능은 이를 이용해, 특정 상황에 적합한 행동을 찾아내도록 한다. 하지만, 새로운 게임에서 개발자의 지식에 의존한 방법은 실패할 가능성이 높기 때문에 게임에 대한 지식에 의존할 필요가 적은 탐색/계획 기법이 필요하다. 본 논문에서는 이런 특성을 가진 다양한 기법 중 MCTS를 이용하였다.

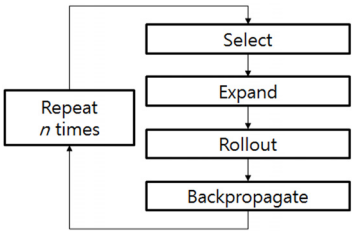


그림 2 Monte-Carlo Tree Search
Fig. 2 Monte-Carlo Tree Search

초기 MCTS는 현재 게임 상태(s_0)에서 시작하여, 무작위로 다양한 행동들을 순차적으로 시뮬레이션 해보고 각 행동의 기대 보상을 추정하는 알고리즘이었지만, 최근 많이 사용되는 UCT (Upper Confidence Tree)는 UCB (Upper Confidence Bound) 알고리즘과 MCTS를 결합하여 기대 보상과 불확실성을 고려하여 효율적으로 탐색한다[4].

그림 2는 MCTS 알고리즘의 흐름을 보여준다. 각 게임 틱마다, 인공지능은 MCTS 알고리즘을 실행하여 현재 상태에 적합한 행동을 결정한다. (1) Select에서는 UCB 알고리즘을 이용해 현재 상태 s_0의 최선의 선택을 결정한다. 그리고 이를 시뮬레이션하여 다음 상태 s_1(s_0의 자식 상태)와 보상 r_1을 구한다. 아직 시도해 보지 못한 행동이 있는 상태 s_l (l 번째 상태)가 나타날 때까지 이 과정을 반복하여 탐색한다. (2) Expand에서는 이전에 행동해보지 못한 행동을 시뮬레이션하여 새로운 게임 상태 s_{l+1}와 보상 r_{l+1}를 찾아낸다. (3) Rollout에서는 새로운 상태 s_{l+1} 이후에 게임 종료 상태 혹은 정해진 깊이까지 무작위로 행동을 결정하여 새로운 게임 상태와 보상을 탐색한다. (4) Backpropagate에서는 시뮬레이션 한 결과인 각각의 게임 상태를 탐색한 횟수와 기대 보상을 지금까지 탐색한 게임 상태에 기록한다. 보통 기대 보상은 특정 게임 상태의 자식 상태의 평균 보상으로 정한다. 정해진 횟수 n만큼 반복한 뒤에는 현재 게임 상태 s_0에 기록된 각각의 행동의 선택횟수와 기대 보상에 따라 다음 행동 a^*를 결정한다.

Select 단계에서 사용한 UCB 알고리즘은 Multi-Armed Bandit 문제에서 탐색(Exploration)과 활용(Exploitation) 딜레마를 해결하기 위한 대표적인 알고리즘이다. UCB 알고리즘 중에 하나인 UCB1은 (1)과 같다. v_i과 n_i는 각각 트리의 i 노드의 기대 보상과 방문 횟수를

의미하며, v_i는 [0,1]사이의 실수 값이다. 또한, N은 현재까지 부모 노드의 총 방문횟수를 뜻하며 C는 조정 가능한 상수이다, 보통 $\sqrt{2}$를 사용한다. 즉, 공식의 왼쪽 부분(v_i)는 기대 보상이 클수록 커지며, 공식의 오른쪽 부분($C\sqrt{\ln N/n_i}$)은 불확실성을 의미한다. 다음 틱에 할 행동으로 가장 UCB1값이 높은 행동을 선택한다.

$$UCB1 = v_i + C\sqrt{\frac{\ln N}{\mathrm{n_i}}} \quad (1)$$

GVG-AI 경진대회에서는 시간(40ms) 때문에 탐색 횟수가 제한되기 때문에, 효율적인 탐색이 가장 중요하다. 기존 UCB1을 이용한 MCTS가 많은 경우 효율적이라고 알려져 있지만, GVG-AI 플랫폼의 일부 게임(예, Zelda)와 같이 보상의 발생 빈도가 매우 적은 경우 MCTS의 탐색은 무작위 탐색과 같아진다. 만약 알고리즘이 몇 단계 미래의 희귀한 보상이 존재하는 게임 상태를 찾더라도, 기존 UCB1에서 이 정보가 무의미할 정도로 축소되어, 보상을 얻을 수 있는 방향으로 충분히 탐색을 하지 않을 가능성이 크다. 본 논문에서는 이 문제를 해결하기 위해 보상이 발생하는 게임 상태 방향에 추가 보상을 주는 GreedyUCB1(2)을 제안한다.

$$GreedyUCB1 = v_i + C\sqrt{\frac{\ln N}{\mathrm{n_i}}} + t_i \quad (2)$$

위 수식에서 t_i는 행동 a_i 에 대한 보상의 총 누적값이다. 이 값에 MCTS를 반복하면서 행동 a_i를 포함한 자식 노드가 받은 보상이 부모 노드에 누적한다. GreedyUCB1의 t_i값만 제외하고 나머지 부분은 UCB1과 같다. GreedyUCB1은 처음에는 t_i의 값이 작아 기존 MCTS와 유사하게 탐색을 하지만, 보상이 발생하는 상태를 찾아 t_i의 값이 커지게 되면, 비록 v_i가 작더라도 t_i이 큰 방향으로 탐색을 하게 된다.

그림 3은 GreedyUCB1과 UCB1를 비교한 예이다. 그림에서 노드 하나는 특정 게임 상태를 나타내며, s_0는 현재 상태 a_0, a_1은 각각의 상태에서 가능한 행동을 나타낸다. s_{21}에서 0.01의 보상이 발생했을 때 UCB1은 현재 깊이의 평균값이 위로 전파되면서 $0.005, 0.0025$와 같이 급격히 감소한다. 더 먼 미래에 보상이 있을수록 더 많이 감소할 것이다. 다음 시뮬레이션에서 s_{10}과 s_{11}의 기대 보상의 차이가 적기 때문에, s_{21} 방향으로 충분히 탐색할 동기가 부족해진다. 반면, GreedyUCB1은 값이 누적되면서 서서히 증가하는 경향을 보이기 때문에 먼 미래에 매우 작은 보상을 받을 것으로 예측되더라도 UCB1에 비해서 s_{21} 방향으로 훨씬 더 많은 탐색을 수행할 것이다.

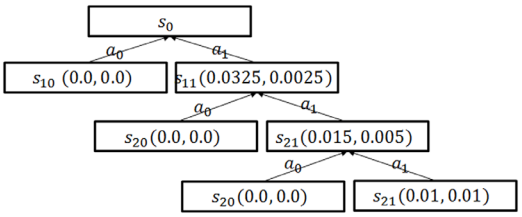


그림 3 GreedyUCB1(왼쪽)과 UCB1(오른쪽)의 비교
Fig. 3 Comparison of GreedyUCB1 (left) and UCB1 (right)

제공되는 게임들 중에는 이동만 가능한 게임이 있는 반면, 공격까지 가능한 게임도 있다. 공격이 가능할 경우, MCTS의 Rollout 방법을 수정하여 공격위주의 플레이를 하도록 하였다. 보통 이런 게임에서는 제한된 시간 안에 다른 개체를 공격하여 제거하거나, 파괴하는 것이 목적이기 때문에, 공격행동이 해당 게임 플레이에서 필수적인 행동이기 때문이다. 기본 Rollout은 Expend 뒤 상태s_{l+1}에서 가능한 행동을 동일한 확률로 무작위로 선택하도록 되어있지만, 제안하는 Rollout 방법은 공격행동에 확률을 높였다. 만약 가능한 행동이 상, 하, 좌, 우, 그리고 공격이라면 원래 Rollout은 모든 행동이 선택될 확률이 1/5이지만, 수정된 Rollout은 공격행동이 선택될 확률이 2/6이고 나머지 행동은 1/6이다. 확률의 수치는 경험적으로 정했다. 이는 비록 특정 게임들에 대한 지식을 사용하고 있지만, 이에 해당하는 게임이 많으며, 그 외의 게임에는 영향을 주지 않기 때문에 부작용은 최소화 할 수 있다.

4. 실험 및 결과

다양한 비디오 게임을 플레이할 수 있는 하나의 인공지능을 연구/실험 하는 것은 매우 어려운 작업이었지만, 최근 ALE와 GVG-AI와 같은 플랫폼이 개발 되면서, 몇몇 연구결과가 발표되었다. 또한 이를 이용해 각종 국제 학술대회에서 인공지능을 구현하여 비교하는 경진대회가 개최되고 있는데, 경진대회에는 다양한 팀에서 개발한 최신 인공지능이 제출된다. 동일한 규칙아래 경쟁하기 때문에, 경진대회 결과는 제안하는 인공지능의 성능을 검증하고 다른 인공지능 기법과 비교하기에 알맞다.

경진대회가 시작하기 몇 달 전에 규칙이 공개되고 참가자들은 그에 맞춰 각자의 인공지능을 개발한다. 이 기간 동안에 훈련용 게임 10개가 공개된다. 참가자들은 이를 이용해 인공지능을 개발하고, 이렇게 개발된 인공지능의 일반 성능을 검증하기 위해 서버에만 있는 검증용 게임 10개를 이용한다. 검증용 게임은 참가자들에게 공개되지 않고, 이를 이용한 결과만 공개된다. 참가자들이 경진대회 서버에 자신의 인공지능을 업로드하면, 동일한

사양의 서버에서 모든 인공지능의 성능을 훈련용 게임과 검증용 게임에서 테스트 한다. 이 결과는 실시간으로 모두에게 공개된다. 최종 경진대회 결과는 훈련용/검증용 게임이 아닌 새로운 10개의 테스트 게임의 플레이 결과에 의해 결정된다. 각각 500게임씩 플레이한다. 각각의 게임에서 점수의 축척이 매우 다르기 때문에, 얻은 점수를 기준으로 순위를 매긴 뒤, F1 그랑프리 레이싱 방식으로 순위에 따라 다시 점수를 부여한다(25, 18, 15, 12, 10, 8, 6, 4, 2 그리고 그 외는 1점). 예를 들어 어떤 게임에서 3점을 얻어 1등을 하면 25점을 받지만, 또 다른 게임에서 100점을 얻어 3등을 한다면 15점을 받는다.

본 논문에서 제안한 방법을 적용한 인공지능은 2014년 IEEE CIG에서 열린 GVG-AI 대회에서 14팀 중 4위의 성적을 거두었다(표 1, 6위까지만 표시, 제안하는 방법은 GreedyUCB1으로 표기). 참가한 전체 인공지능의 평균 획득 점수는 60.71점이며, 22.9% 승률(500게임 플레이)을 기록했다. 제안한 방법으로 개발한 인공지능은 68점을 획득하였다. 이 점수는 전체 게임에 대해 F1 그랑프리 식으로 종합한 점수이다.

5위권 안의 대부분의 인공지능이 훈련용 게임 점수보다, 테스트용 게임 점수에서 약 20-30점이 하락했다(표 2). 이런 현상은, 공개된 특정 게임에서 최적화 되도록 설계된 인공지능이 다른 규칙을 가진 새로운 게임에 적합하지 않아 발생할 수 있다. 반면 Shnokin과 제안하는 방법으로 구현한 인공지능은 약 10점 상승하면서 순위가 향상되었다. 즉, 점수가 하락한 다른 인공지능보다 새로운 환경에 더 잘 적응했다고 볼 수 있다.

표 1 CIG 2014 GVG-AI 대회 결과
Fig. 1 CIG 2014 GVG-AI competition results

Rank	AI	Score	Win. ratio
1	Adrienctx	158	51.2%
2	JinJerry	148	43.2%
3	Shmokin	77	31.6%
4	**GreedyUCB1**	**68**	**25.4%**
5	culim	61	24.8%
6	MMbot	59	26.0%
...	...	...	...
Avg.		60.71	22.9%

표 2 훈련/평가과정에서 점수 변화(인공지능과 점수)
Fig. 2 Ranks changes between training and testing (AI and points)

Rank	Training Games	Testing Games
1	adrienctx (191)	adrienctx (158)
2	JinJerry (174)	JinJerry (148)
3	MnMCTS (96)	Shmokin (77)
4	MMbot (83)	**GreedyUCB1(68)**
5	Shmokin (66)	culim (61)
6	**GreedyUCB1 (56)**	MMbot (59)
...	...	...

제안하는 방법으로 구현한 인공지능은 기본 MCTS보다 테스트 게임들 중에서 Butterflies와 Zelda에서 더 높은 성능을 보였다. Butterflies는 시간제한을 제외하면 게임 패배 조건이 없어 위험부담이 적고, 적당한 위치로 이동하기만 하면 추가 보상을 얻기 쉬워 빠른 탐색에 유리한 GreedyUCB1이 기존 UCB1에 비해 유리하며, Zelda는 보상이 드물지만, 위협이 되는 NPC를 제거하기만 하면 게임 플레이가 상대적으로 쉬워지기 때문에 제안한 rollout 방법에 적합했다. 반면, Portals과 Sokoban처럼 빠른 탐색보다 정확한 탐색이 필요한 게임에 대해서는 기본 MCTS보다 낮은 성능을 보였다.

5. 결론 및 향후 연구

본 논문에서는, 최근 활발히 연구되고 있는 MCTS기반 인공 일반지능에 적용할 수 있는 GreedyUCB1과 게임 분석을 통해 얻은 지식을 이용한 rollout 방법을 제안했다. GVG-AI 플랫폼에서 기존 MCTS는 뛰어난 성능을 보여주지만, 게임에 따라 탐색 효율이 급격히 나빠지는 경우가 발생한다. 제안한 Greedy-UCB1은 기존 MCTS의 UCB1을 대체한다, 작은 보상에 가중치를 부여하여, 유망한 게임 상태를 집중적으로 탐색하도록 한다. 추가로, 제시된 rollout 방법은 공격 행동이 일부 게임에서, 중요한 행동이라는 점에 착안하여 공격확률을 높인다.

제안한 방법을 적용한 인공지능은 2014년 IEEE CIG의 GVG-AI 경진대회에서 4위의 성적을 거두었다. 제안한 방법은 몇몇 게임들에 대해서는 기존 MCTS보다 좋은 성능을 보이고 있지만, 또 다른 게임에서는 낮은 성능을 보이고 있다. 이는 다음과 같은 문제 때문에 발생한다. 첫째, GreedyUCB1은 적은 보상을 강화시켜 이용하고 있지만, 만약 탐색을 통해 전혀 보상을 찾지 못하면 이는 큰 효과를 보지 못한다. 둘째, 신속한 탐색보다 정확한 탐색이 필요한 게임들의 경우 오히려 불리할 수 있다. 특히 Portals와 Sokoban과 같은 퍼즐 게임은 넓은 범위에 대해 정확한 탐색이 필요하다, 하지만 GreedyUCB1과 같은 기법은 탐색 속도 향상을 위한 추가 보상을 이용하기 때문에 정확한 탐색이 힘들 수 있다. Sokoban 같은 게임에서 초반의 잘못된 행동은 되돌릴 수 없기 때문에 게임의 승리에 치명적인 영향을 줄 수 있다.

References

[1] M. Genesereth, N. Love, and B. Pell, "General Game

Playing: Overview of the AAAI Competition," *AI Mag.*, Vol. 26, No. 2, p. 62, Mar. 2005.
[2] J. Levine, C. B. Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, and T. Thompson, "General Video Game Playing," *Artif. Comput. Intell. Games*, Vol. 6, pp. 77-83, Dec. 2013.
[3] D. Perez, S. Samothrakis, J. Togelius, T. Schaul, and S. Lucas (2014), The GVG-AI Competition [Online]. Available: http://www.gvgai.net/ (downloaded 2015, Mar. 15)
[4] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Trans. Comput. Intell. AI Games*, Vol. 4, No. 1, pp. 1-43, Mar. 2012.
[5] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Mach. Learn.*, Vol. 47, No. 2-3, pp. 235-256, May 2002.
[6] GGP Homepage, [Online]. Available: http://games.stanford.edu/(downloaded 2015, Feb. 28)
[7] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An Evaluation Platform for General Agents," *J. Artif. Intell. Res.*, Vol. 47, pp. 253-279, Jun. 2013.
[8] M. Gendron-Bellemare, "Fast, Scalable Algorithms for Reinforcement Learning in High Dimensional Domains," University of Alberta, 2013.
[9] M. Hausknecht, P. Khandelwal, R. Miikkulainen, and P. Stone, "HyperNEAT-GGP: A HyperNEAT-based Atari General Game Player," *Proc. of the 14th Int. Conf. on Genetic and Evol. Comput. Conf.*, pp. 217-224, Jul. 2012.
[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *Proc. of the Deep Learning. Neural Information Processing Systems Workshop*, Dec. 2013.
[11] T. Schaul, "An Extensible Description Language for Video Games," *IEEE Trans. Comput. Intell. AI Games*, Vol. 6, No. 4, pp. 325-331, Oct., 2014.
[12] S. M. Lucas, S. Samothrakis, and D. Perez, "Fast Evolutionary Adaptation for Monte Carlo Tree Search," *Application of Evolutionary Computation*, pp. 349-360, Springer, Berlin, 2014.
[13] D. Perez, S. Samothrakis, and S. Lucas, "Knowledge-based Fast Evolutionary MCTS for General Video Game Playing," *Proc. of the 2014 IEEE Conf. on Comput. Intell. and Games (CIG)*, pp. 1-8, Oct., 2014.

박 현 수
2013년 세종대학교 컴퓨터공학과 졸업(석사). 2013년~현재 세종대학교 컴퓨터공학과 박사과정. 관심분야는 게임 인공지능, 진화연산, 강화학습

김 현 태
2015년 세종대학교 컴퓨터공학과 졸업(석사). 2015년~현재 (주)와이즈넛 성장기술본부 연구원. 관심분야는 진화연산, 게임인공지능 등

김 경 중
2007 연세대학교 컴퓨터과학과 졸업(박사)
2007년~2009년 코넬대학교 박사후연구원
2009년~현재 세종대학교 컴퓨터공학과 부교수. 관심분야는 진화연산, 로봇 지능, 게임 지능, 3차원 프린터