Journal of Korea Game Society 2015 Oct; 15(5): 109-118
http://dx.doi.org/10.7583/JKGS.2015.15.5.109

게임 프로그래밍

삼목 게임에서 최상의 첫 수를 구하기 위해 적용된 신뢰상한트리 알고리즘

이병두, 박동수, 최영욱
세한대학교 체육학부
blee026@korea.com, {pds, cyw}@sehan.ac.kr

The UCT algorithm applied to find the best first move in the game of Tic-Tac-Toe

Byung-Doo Lee, Dong-Soo Park, Young-Wook Choi
Division of Sports Science, Sehan University

요 약

고대 중국에서 기원된 바둑은 인공지능 분야에서 가장 어려운 도전 중의 하나로 간주된다. 지난 수년에 걸쳐 MCTS를 기반으로 하는 정상급 컴퓨터바둑 프로그램이 놀랍게도 접바둑에서 프로기사를 물리쳤다. MCTS는 게임이 끝날 때까지 일련의 무작위 유효착수를 시뮬레이션 하는 접근법이며, 기존의 지식기반 접근법을 대체했다. 저자는 MCTS의 변형인 UCT 알고리즘을 삼목 게임에 적용하여 최선의 첫 수를 찾고자 했으며, 순수 MCTS의 결과와 비교를 했다. 아울러 UCB 이해를 위한 다중슬롯머신 문제를 풀기 위해 엡실론-탐욕 알고리즘과 UCB 알고리즘을 소개 및 성능을 비교하였다.

ABSTRACT

The game of Go originated from ancient China is regarded as one of the most difficult challenges in the filed of AI. Over the past few years, the top computer Go programs based on MCTS have surprisingly beaten professional players with handicap. MCTS is an approach that simulates a random sequence of legal moves until the game is ended, and replaced the traditional knowledge-based approach. We applied the UCT algorithm which is a MCTS variant to the game of Tic-Tac-Toe for finding the best first move, and compared it with the result generated by a pure MCTS. Furthermore, we introduced and compared the performances of epsilon-Greedy algorithm and UCB algorithm for solving the Multi-Armed Bandit problem to understand the UCB.

Keywords : Go(바둑), Tic-Tac-Toe(삼목), MCTS(몬테카를로 트리탐색), UCT(신뢰상한트리), Multi-Armed Bandit(다중슬롯머신), epsilon-Greedy(엡실론-탐욕), UCB(신뢰상한)

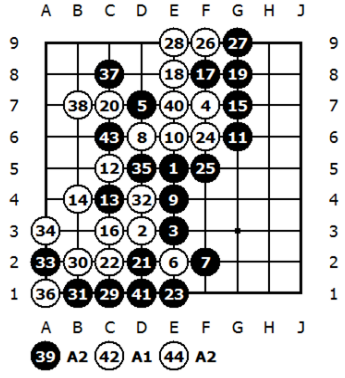
Received: Sep 09, 2015 Accepted: Oct. 16, 2015
Corresponding Author: Byung-Doo Lee (Sehan University)
E-mail: blee026@korea.com

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

2,500년 전 중국에서 기원된 2인-제로섬-완전정보게임인 바둑은 아직까지 컴퓨터가 완전 제압을 못하고 있는 마지막 게임이다[1,2,3,4]. 1990년대부터 컴퓨터 바둑의 기력 향상을 위해 전 세계적으로 많은 인공지능 연구자들은 최신 인공지능 기법을 적용하였으나, 그 기력이 프로기사에 못 미쳐 새로운 방법을 고대하고 있었다[4,5,6,7,8].

대안책으로 몬테카를로 트리탐색(MCTS: Monte-Carlo Tree Search)을 바둑에 적용한 결과, 마침내 2007년에 MoGo라는 컴퓨터 바둑이 [Fig. 1]에서 보듯이 9줄바둑에서 프로기사인 Guo Juan 5단을 제압했다. 또한 2009년에는 Fuego라는 컴퓨터 바둑이 최정상 기사인 Zhou Junxun 프로 9단을 누르면서 9줄바둑에서 컴퓨터가 인간을 완전히 제압했다[4,6,9].



[Fig. 1] The first computer Go won against a pro Go player

또한 19줄바둑에서도 컴퓨터 바둑은 선전을 해왔으며, 특히 2008년에 MoGo라는 컴퓨터 바둑은 Kim Myungwan 프로 8단을 9점 접바둑으로 이겼고, 최근에는 [Table 1]에서 보듯이 4점 접바둑으로 프로기사를 제압하고 있어 상당한 수준의 아마 기력을 보유하게 되었다[2,4,6,9]. 참고로 MCTS 바둑이 등장하기 전까지 최강의 컴퓨터 바둑 프로그램은 KGS(Kiseido Go Server)가 6급을 인정한 Many Faces of Go이였으나, 근간에 MCTS 바둑 기력이 해마다 향상되어 현재 최강의 컴퓨터 바둑 프로그램은 2012년에 아마 6단을 인정받은 Zen이다[9,10,11].

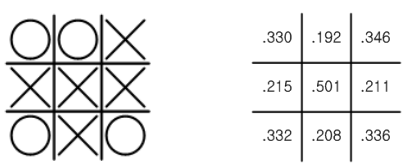
[Table 1] The first computer Go programs won against professional Go players[2,4,6,11,12]

Year	Handicap	Human level	Computer program
2008	9	8 dan	MoGo
2008	8	4 dan	Crazy Stone
2008	7	4 dan	Crazy Stone
2009	7	9 dan	MoGo
2009	6	1 dan	MoGo
2010	6	4 dan	Zen
2012	5	9 dan	Zen
2013	4	9 dan	Crazy Stone

2. 개요 및 관련 연구

2.1 삼목 게임

삼목 게임(Tic-Tac-Toe 또는 Noughts and Crosses)은 전 세계적으로 잘 알려진 게임 중의 하나이며, 두 대국자가 번갈아가며 3×3칸의 종이 위에 X와 ○를 번갈아 연필로 써서 가로, 세로 또는 대각선상에 동일한 모양이 연속하여 3개가 형성되면 이기는 게임이다[13]. 참고로 [Fig. 2](a)는 첫 번째 대국자인 X가 가로 방향으로 승리를 한 장면이다.



(a) (b)

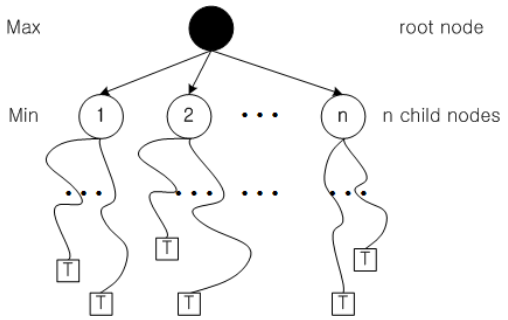
[Fig. 2] (a) A finished game won by the first player, X and (b) winning rates of each position[1,2,4,6]

2.2 몬테카를로 트리탐색

게임프로그래밍에서 대부분의 성공적인 접근 방법은 평가함수(evaluation function)를 이용한 게임트리탐색이다. 이 접근법은 평가함수를 통해 계산된 평가값을 갖고 현재 게임자의 좋은 착수위치를 계산해내는 방법이다. 그러나 아직까지 컴퓨터 바둑에서 이러한 실질적인 평가함수를 구축하는 것은 어렵다.

컴퓨터 게임에서 게임트리 내 게임 깊이와 분기수가 비교적 작은 경우에는 최소최대탐색 또는 음부호최대탐색을 수행하여 컴퓨터 게임 프로그램의 성능을 비약적으로 향상시킬 수가 있으나, 이로 인해 게임트리의 규모가 기하학적으로 증가하는 약점이 있다. 이러한 기하학적인 게임트리의 증가를 막기 위해 효율적인 가지치기 기법인 알파-베타 가지치기가 제시되어 왔으나 바둑과 같이 게임 깊이와 분기수가 매우 큰 경우에는 이를 이용한 전역탐색(exhaustive search)을 수행할 수가 없다. 결국 이러한 난점을 극복하기 위해 제안된 것이 몬테카를로(MC) 시뮬레이션이다.

MC 시뮬레이션 방법은 하위트리를 무작위로 표집하여 시행할 행위들에 대한 값어치를 추정해내는 기법이다. 물론 추정값의 정확도는 시뮬레이션의 횟수에 좌우되며, 시뮬레이션의 횟수가 클수록 추정값은 더 정확해진다[14]. [Fig. 3]은 단순한 확률적 트리탐색을 위해 MC 시뮬레이션을 통한 평가가 어떻게 이루어지는지를 보여주고 있다.



[Fig. 3] Greedy algorithm for MC evaluation

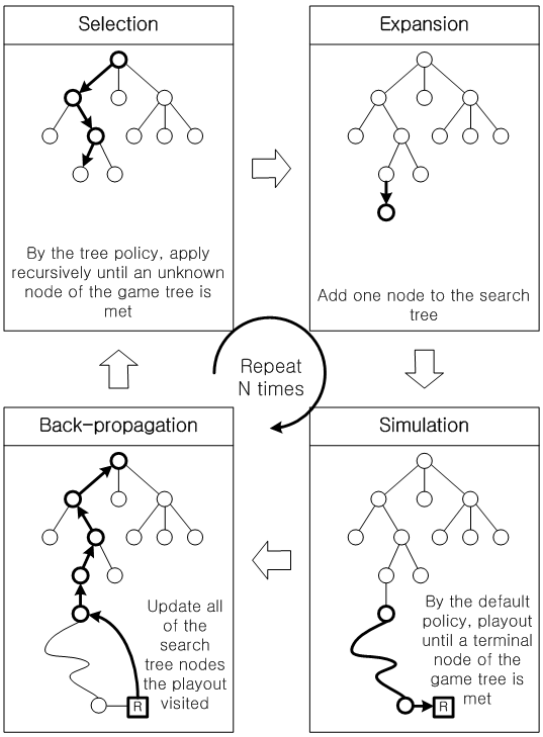
우선 뿌리노드를 확장한 후, 행위가 가능한 n개의 자식노드를 트리에 첨가한다. 이후 단말노드 T에 이를 때까지 시뮬레이션을 통해 각 자식노드에 대해 값어치를 추정한다. 각 자식노드에 대한 값어치는 시뮬레이션의 합성값 또는 시뮬레이션 결과값들의 평균을 각 자식노드의 추정값으로 적용할 수 있다. 그러고 나서 각 자식노드들의 추정값 중 최댓값을 갖는 자식노드를 선택하여 게임 행위를 할 수 있다.

MC 시뮬레이션은 게임트리의 엄청난 분기로 인해 생겨난 노드들을 전역탐색해야 하는 단점을 해결해 주나, 어떠한 방식으로 연속되는 게임의 행위를 시뮬레이션 해야 하는 가에 대한 문제가 있다[14]. 이 문제를 해결하기 위한 한 예로 단순 탐욕적인 MC 시뮬레이션을 정제한 형태인 점진적 가지치기(progressive pruning)가 있다.

점진적 가지치기에서 각 행위는 그에 따른 평균 μ와 표준편차 σ를 갖는다. 시뮬레이션을 통해 이 값들을 추정 및 변경을 하여 생성된 통계량을 갖고 현재의 행위가 다른 행위에 비해 일정 신뢰도(confidence) 내에서 우월한지 여부를 통계학적으로 쉽게 알아낼 수 있다. 점진적 가지치기는 다른 행위에 비해 확실히 우월한 행위에 대해서는 더 이상 시뮬레이션을 하지 않게 된다. 즉 시뮬레이션에 걸리는 시간을 절약하여 좀 더 유망할 것 같은 행위에 좀 더 많은 시간 투자를 하게 된다.

MCTS는 위치평가함수(position evaluation function)를 필요로 하지 않는 최대우선탐색(best-first search) 방법으로 탐색공간에 대한 무작위적 탐험(randomized exploration)에 근간을 두고 있으나[1,6,15], 바둑과 같은 게임에서 중시하는 전략과 전술을 적절히 적용할 수 없는 약점을 안고 있다[1,4,16].

기본적인 MCTS 알고리즘은 [Fig. 4]에서 보듯이 선택, 확장, 시뮬레이션, 역전파라는 4단계로 구성되어 있으며[6,17,18], 알고리즘을 수행하기 위해서는 (1)게임 진행시 발생된 행동들에 대한 정보를 담는 게임트리(game tree)와 (2)시뮬레이션을 위해 사용되는 정보를 담는 탐색트리(search tree)가 필요하다[2,14].



[Fig. 4] Four steps for UCT algorithm[2,6,19,20]

-선택(Selection): 1회전마다 트리정책(tree policy 또는 selection policy)을 반영하여 게임트리 내 확장이 안 된 노드들이나 단말노드를 만날 때까지 반복하여 노드를 선택한다.

-확장(Expansion): 만약 선택된 노드가 탐험이 안 된 노드인 경우에는 탐색트리에 추가된다.

-시뮬레이션(Simulation): 디폴트정책(default policy 또는 simulation policy)에 의해 수행되며, 탐색트리의 단말노드로부터 시뮬레이션을 시작하여 게임트리의 단말노드를 만날 때까지 시뮬레이션을 계속한다.

-역전파(Back-propagation): 시뮬레이션의 결과를 선택된 노드로부터 탐색트리의 뿌리노드까지 역전파하며 경유되는 각 노드에 대해 통계적 수치 값을 갱신해 나간다.

신뢰상한트리(UCT: Upper Confidence Bound applied to Trees)는 MCTS의 변형이며, 게임트리 내의 행동은 신뢰상한(UCB: Upper Confidence Bounds)이라는 정책을 따른다[17]. 이를 달리 말하면 [Fig. 4]에서 보듯이 선택과정에 있는 트리정책을 위해 UCB를 사용하며, 시뮬레이션 과정에 있는 디폴트정책을 위해 MC 시뮬레이션을 사용한다.

2.3 다중슬롯머신 문제

UCB를 활용하기 위해서는 우선적으로 다중슬롯머신(MAB: Multi-Armed Bandit) 문제를 이해해야 한다. 다중슬롯머신 문제는 취할 수 있는 행동이 많이 존재하고 각각의 행동을 통해 얻게 될 보상에 대한 불완전한 정보를 갖은 상태에서, 보상을 극대화하기 위해 어떤 행동을 해야 될지를 추론하는 수학적 모델이다[21].

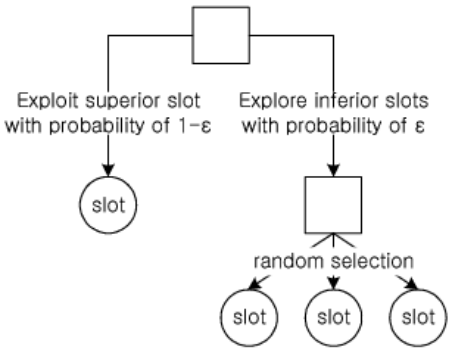
바둑뿐만 아니라 대부분의 컴퓨터 게임은 제한된 시간에 최적의 해를 요구하고 있다. 순수 몬테카를로 알고리즘(pure MC algorithm)은 행동할 수 있는 곳에 거의 균등한 정도의 시뮬레이션을 수행한 후 보상값이 가장 높은 행동을 취하는 알고리즘이다. 즉 여러 번의 시뮬레이션을 통해 보상값이 가장 높은 행동을 취하는 최대우선탐색 알고리즘으로 탐험(exploration)보다는 활용(exploitation)에 기반을 두고 있다.

그러나 현실에서는 보상값이 높은 행동에 더 많은 시간 투자를 하고 싶고, 가끔은 탐험을 통해 더 좋을 수 있는 행동을 찾고자 한다. 이 문제를 탐험-활용 딜레마(Explore-Exploit dilemma)라고 부르며[20,21], 이를 해결하는 알고리즘 중 가장 간단한 것은 엡실론-탐욕(epsilon-Greedy) 알고리즘이며, 가장 많이 사용하고 있는 것은 UCB 알고리즘이다.

2.4 엡실론-탐욕 알고리즘

탐험-활용 딜레마를 해결하는 가장 간단한 엡실론-탐욕 알고리즘은 ϵ만큼 탐험을 수행하고, $1-\epsilon$만큼 활용을 수행하는 알고리즘이다. 즉 [Fig. 5]에서 보듯이 가장 성공률이 높은 행동을

기본으로(실제로는 $1-\epsilon$의 확률로) 수행하고, 가끔씩은(실제로는 ϵ의 확률로) 현재 최고의 성공률을 보이지 않는 행동을 무작위로 수행하는 알고리즘이 된다[20].



[Fig. 5] epsilon-Greedy algorithm for exploitation and exploration

엡실론-탐욕 알고리즘은 탐험과 활용을 수행하는 장점이 있으나, 문제점으로는 인위적으로 ϵ값을 고정시켰기 때문에 성공률이 저조한 행동을 강제적으로 탐험을 해야 하는 과도탐험이 발생하며, 또한 수행 초기에 더 많은 탐험을 해야 함에도 불구하고 탐험을 할 수 없다는 약점을 안고 있다[21].

2.5 신뢰상한 알고리즘

탐험-활용 딜레마 해결을 위해 가장 많이 사용되고 있는 UCB 알고리즘은 무작위로 행동을 선택하는 것이 아니라, 수행해왔던 행동에 대해 얼마나 유용한지를 판단하여 활용과 탐험의 균형을 유지하는 알고리즘으로 특징은 다음과 같다[20,22].

-행동의 선택에 있어 현재 가장 높은 UCB 값을 갖는 행동을 취한다.

-이후 가끔씩 탐험을 결정하고 현재 최적처럼 보이지 않는 행동을 시행한다.

-시간이 지남에 따라 UCB 값이 가장 높은 행동을 많이 시행하게 된다.

[23]이 제안한 여러 개의 UCB 알고리즘 중 가장 많이 사용하는 알고리즘이 UCB1 알고리즘이다. UCB1 알고리즘은 게임 트리에서 어떤 자식노드를 취해 확장해 나가고 MC 시뮬레이션을 할 것인가를 (eq. 1)에 있는 UCB1 값을 사용하여 결정하게 된다[9,17,19,20,23,24].

$$UCB1 = \overline{X_i} + C\sqrt{\frac{2\ln n}{n_i}} \qquad \text{(eq. 1)}$$

(eq. 1)에 있는 UCB1 값은 활용부과 탐험부의 합으로 구성되어 있으며, 활용부에 있는 $\overline{X_i}$는 현재 검사된 노드 i로부터의 평균 보상값이 되며, 탐험부에 있는 상수값 C는 탐험바이어스(exploration bias), n은 i노드의 부모노드를 방문한 횟수, n_i는 노드 i를 방문한 횟수가 된다.

2.6 수행된 관련 연구

저자는 본격적인 컴퓨터 바둑 제작에 앞서 최신의 인공지능 기법인 MCTS와 유전 알고리즘(Genetic algorithm)의 활용 가능성 정도를 파악하기 위해 이들을 삼목 게임에 적용했으며, 적용한 결과 밝혀진 주요 결과는 다음과 같이 요약된다.

■ 삼목 게임에서는 [Fig. 2](a)에서 보듯이 첫 수로 9곳을 착수할 수 있다. 착수 가능한 9곳 중 승률이 가장 높은 첫 수 위치를 찾아내기 위해 100,000번의 MC 시뮬레이션을 수행한 결과, [Fig. 2](b)에서 보듯이 첫 수로 중앙에 둔 경우 50.1%, 귀에 둔 경우 33.6%, 변에 둔 경우 20.6%의 평균 승률을 보여, 첫 수로 "중앙이 우선, 귀가 다음, 변은 나중"이라는 사실을 최초로 밝혀냈다[1,2,6].

■ 또한 삼목 게임에 있어 두 대국자의 최선의 수순과 게임 결과를 알아보기 위해 MCTS를 적용하여 [Fig. 6](a)와 같은 최선의 수순을 보였으며, 아울러 인간과 컴퓨터 간의 최선의 게임결과는 무승부가 됨을 보였다[6]. 결국 MCTS를 삼목 게임에 적용하게 되면 인간과의 대국에서 컴퓨터는 절대로 지지 않는다는 것을 보였다[4,6].

[Fig. 6] (a) The best move sequence with MC simulation; (b) and (c) are exemplary best and worst move sequences generated by MCTS with strategy fitness function, respectively[1,4]

■ 바둑은 전략과 전술을 중요시하게 다루는 게임이나 MCTS에서는 이를 적절히 다룰 수 없는 약점을 안고 있다. 이를 보완하기 위하여 전략기반 유전 알고리즘(strategy-based genetic algorithm)을 MCTS에 적용한 결과 삼목 게임에서 전략과 전술을 구사할 수 있음도 보였다[1,16,25,26]. 참고로 [Fig. 6](b)는 전략함수를 이용하여 MCTS에 의해 생성된 최선의 수순 중의 한 예이며, [Fig. 6](c)는 최악의 수순 중의 한 예가 된다.

3. 실 험

3.1 실험 1: 다중슬롯머신 문제

우선 다중슬롯머신(MAB) 문제를 통해 엡실론-탐욕 알고리즘과 UCB 알고리즘이 어떻게 작동되는가를 살펴보자. 한 예로 카지노 객장에 각각 25.0%, 12.5%, 50.0%의 당첨 확률을 갖는 3개의 슬롯 A, B, C가 있다고 가정하고, 손님은 각 슬롯에 대한 당첨 확률을 모르는 상태에서 어떤 슬롯을 어느 정도 베팅을 할 것인가를 살펴보자.

엡실론-탐욕 알고리즘을 위해 $\epsilon = 0.1$로 고정시켰으며, UCB 알고리즘을 위한 UCB1 값은 (eq. 1)을 적용하였으며 상수값 $C = 1$로 고정시켰다.

[Fig. 7]에서 보듯이 엡실론-탐욕 알고리즘은 총 1,000번의 반복 수행을 통해 895번의 활용과 105번의 탐험을 했으며, 슬롯 A에 대해서는 28.6%(63번 방문), 슬롯 B에 대해서는 10.7%(56번 방문), 슬롯 C에 대해서는 48.5%(881번 방문)의 성공률을 추정하고 있다.

```
During 1000 iterations, the numbers of exploitation
and exploration are 895 and 105, respectively.

Win rate of slot A: 0.285714 with 63 tries
Win rate of slot B: 0.107143 with 56 tries
Win rate of slot C: 0.484677 with 881 tries
```

[Fig. 7] Expected success rates with epsilon-Greedy algorithm

반면에 [Fig. 8]에서 보듯이 UCB 알고리즘은 슬롯 A에 대해서는 25.5%(98번 방문), 슬롯 B에 대해서는 11.5%(52번 방문), 슬롯 C에 대해서는 50.7%(850번 방문)의 성공률을 추정하고 있다.

```
Win rate of A: 0.255102 with 98 tries
Win rate of B: 0.115385 with 52 tries
Win rate of C: 0.507059 with 850 tries
```

[Fig. 8] Expected success rates with UCB algorithm

참고로 [Fig. 9]는 순수 MC 알고리즘을 통해 얻어낸 결과이다.

```
Win rate of A: 0.236152 with 343 tries
Win rate of B: 0.115152 with 330 tries
Win rate of C: 0.489297 with 327 tries
```

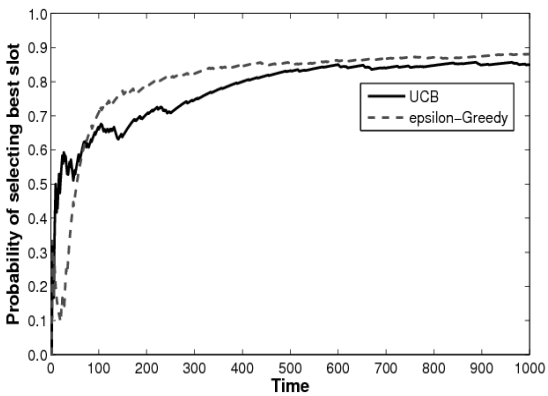
[Fig. 9] Expected success rates with MC algorithm

성공률을 예측하는 정도는 세 알고리즘이 유사하나, 큰 차이로는 [Table 2]에서 보듯이 엡실론-탐욕 알고리즘과 UCB 알고리즘과는 달리 순수 MC 알고리즘은 각 슬롯 A, B, C에 대해 거의 균등한 정도(슬롯 A에 대해 343번, 슬롯 B에 대해 330번, 슬롯 C에 대해 327번)로 방문을 한다는 것이다.

한편 [Fig. 10]은 가장 성공률이 높은 슬롯 C(50%의 성공률)에 대해 엡실론-탐욕 알고리즘과 UCB 알고리즘이 반복 수행 중 얼마나 자주 슬롯 C를 선택했는가를 보여주고 있다.

[Table 2] Comparison of expected success rates among the three algorithms

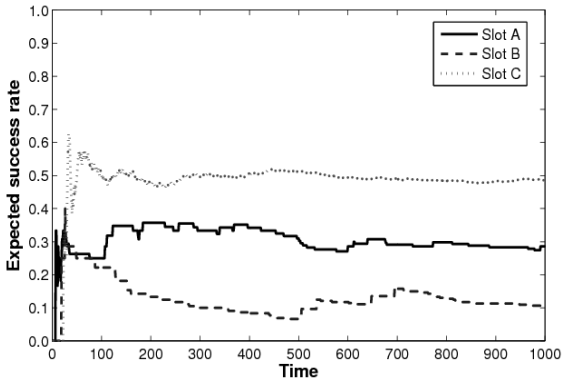
Algorithm	Slot	# of visit	Expected success rate
epsilon-Greedy	A	63	28.6%
	B	56	10.7%
	C	881	48.5%
UCB	A	98	25.5%
	B	52	11.5%
	C	850	50.7%
pure MC	A	343	23.6%
	B	330	11.5%
	C	327	48.9%



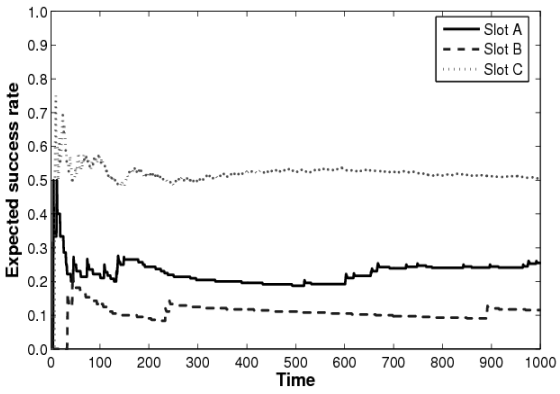
[Fig. 10] Probability of selecting the best slot C

[Fig. 10]에서 보듯이 두 알고리즘은 수행 횟수가 증가할수록 점차 수렴(슬롯 C를 선호)하는 경향은 같으며, 엡실론-탐욕 알고리즘이 UCB 알고리즘보다 수렴 속도가 좀 더 빠르다는 것을 알 수 있다. 또한 엡실론-탐욕 알고리즘은 활용을 위해 현재의 성공률이 높은 슬롯에 집착하는 반면에, UCB 알고리즘은 현재의 성공률이 높은 슬롯에 대한 활용을 고수하면서 성공률이 낮은 슬롯에도 탐험을 지속하고 있음을 보여주고 있다.

[Fig. 11]과 [Fig. 12]는 엡실론-탐욕 알고리즘 및 UCB 알고리즘을 이용하여 수행 횟수에 따른 3개의 슬롯 A, B, C에 대한 성공확률을 예측하는 그림들이 된다.



[Fig. 11] Expected success rates with epsilon-Greedy algorithm



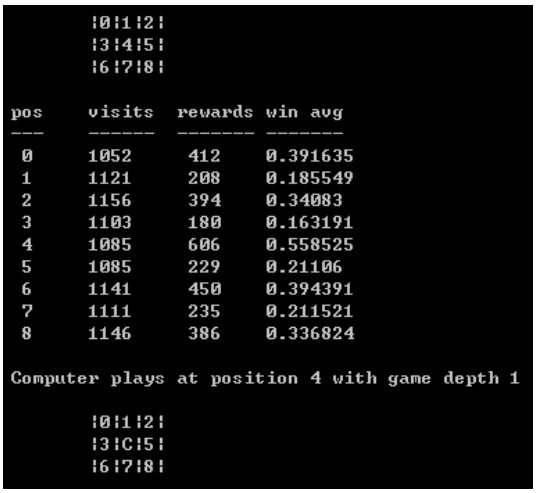
[Fig. 12] Expected success rates with UCB algorithm

[Fig. 11], [Fig. 12]에서 보듯이 대략 300번의 수행 이후부터 점차 수렴하는 것을 볼 수 있으며, 엡실론-탐욕 알고리즘은 슬롯 A, B, C에 대해 각각 최종 28.6%, 10.7%, 48.5%의 성공률을 예측하고 있고, UCB 알고리즘은 슬롯 A, B, C에 대해 각각 최종 25.5%, 11.5%, 50.7%의 성공률을 예측하고 있다. 결국 3개의 슬롯 A, B, C를 갖고 수행한 다중슬롯머신 문제에 대한 성공확률을 예측하는 예에서는 두 알고리즘 간에 큰 차이가 없어 보인다.

3.2 실험 2: 삼목 게임에서의 첫수

[2]에서 보였듯이 순수 MCTS를 적용하여 삼목

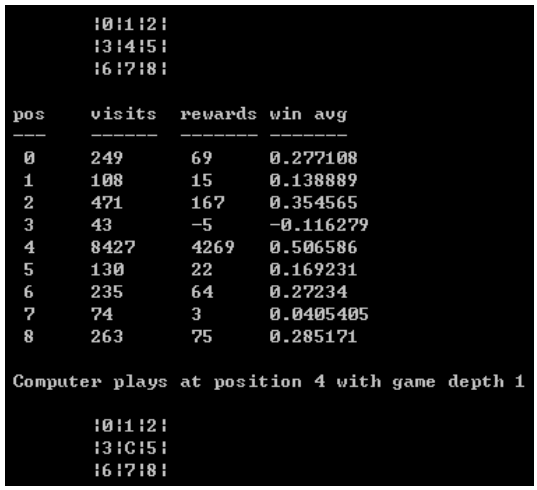
게임에서의 최선의 첫 수를 구해보면, 중앙의 곳이 가장 성공률이 높아 첫 수로 중앙에 두어야함을 보였다. 참고로 [Fig. 13]은 순수 MCTS를 적용하여 10,000번의 시뮬레이션 결과 삼목게임의 첫 수로 둘 수 있는 9곳의 보상값과 성공률을 보여주고 있다.



[Fig. 13] Winning rates of each position generated by pure MCTS algorithm

[Fig. 13]에서 보듯이 9곳의 성공률 중 중앙의 곳인 4의 위치에 대한 성공률이 55.9%로 가장 크기 때문에 첫 수로 중앙에 착수를 하게 된다. 또한 순수 MCTS 알고리즘에 의해 생성된 9곳의 위치를 방문하는 횟수를 살펴보면 거의 균등하게 방문(최소 1,052번~최대 1,156번 방문)함을 알 수 있다.

반면에 [Fig. 14]에서 보듯이 UCB1을 이용한 UCT 알고리즘에 의해 10,000번의 시뮬레이션 결과 방문되는 횟수를 살펴보면 성공률이 높은 곳을 많이 방문(4의 곳을 8,427번)하고, 성공률이 낮은 곳을 적게 방문(3의 곳을 43번)함을 알 수 있다. 이는 UCT 알고리즘의 강점인 탐험과 활용을 적절히 균형을 맞추면서 최고 50.7%의 성공률을 보이는 4의 곳, 즉 중앙의 곳을 첫 수로 결정하는 것을 알 수 있다.



[Fig. 14] Winning rates of each position generated by UCT algorithm

4. 결론 및 제언

4.1 결론

근간에 MCTS를 활용한 컴퓨터 바둑은 급속도로 발전하여 19줄바둑에서 4점 접바둑으로 프로기사를 제압하고 있다. MCTS는 위치평가함수를 사용하지 않는 최대우선탐색으로 탐색공간에 대한 무작위 탐험을 통해 근사적인 최상의 해를 찾아내는 장점이 있으나[1,19], 시뮬레이션에 전적으로 의존하는 MCTS는

-적절한 게임 전략을 구사할 수 없으며[1],

-게임 시간이 제한되는 경우 많은 시뮬레이션 시간으로 인해 최적의 해를 찾아낼 수 없는 단점도 안고 있다.

반면에 MCTS를 변형한 UCT는 탐험과 활용에 대한 균형을 유지하면서 성공률이 높은 행동에 대해 많은 시간을 할애하고 있다. 이는 바둑과 같은 게임에서 중요한 의미를 갖고 있다. 즉 바둑에서는 좋은 수(手)에 대해 많은 시간을 갖고 깊은 수읽기를 요구하고 있다. 결국 UCT를 근간으로 한 컴퓨터 바둑은 순수 MCTS를 근간으로 하는 컴퓨터

바둑보다 우수하다.

저자는 본 논문에서 다중슬롯머신 문제를 해결하기 위해 엡실론-탐욕 알고리즘과 UCB 알고리즘의 성능 차이를 보였으며, 아울러 삼목 게임을 이용하여 순수 MCTS 알고리즘과 UCT 알고리즘의 차이점을 비교하였으며 삼목 게임에서의 최상의 첫수도 제시하였다.

4.2 제언

바둑은 전략과 전술을 중시하는 게임이다. 순수 MCTS 알고리즘과 UCT 알고리즘은 전략과 전술을 다룰 수 없는 약점이 있다. 이를 보완하기 위하여 향후 UCT 알고리즘에 전략기반 유전 알고리즘을 접목하는 연구도 이루어져야 한다. 19줄바둑에 이를 곧바로 적용하는 것은 매우 방대하고 난해한 작업이 된다. 대신 9줄바둑에서 이를 적용한 후 이미 개발된 여타 컴퓨터 바둑과의 성능을 비교하는 것도 좋은 연구과제가 될 듯하다.

REFERENCES

[1] B.D. Lee, “Analysis of Tic-Tac-Toe Game Strategies using Genetic Algorithm”, Journal of Korea Game Society, Vol. 14, No. 6, pp. 39-48, 2014.

[2] B.D Lee, “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.

[3] B.D. Lee and J.W. Park, “Applying Principal Component Analysis to Go Openings”, Journal of Korea Game Society, Vol. 13, No. 2, pp. 59-70, 2013.

[4] B.D. Lee, “Evolutionary neural network model for recognizing strategic fitness of a finished Tic-Tac-Toe game”, Journal of Korean Society for Computer Game, Vol. 28, No. 2, pp. 95-101, 2015.

[5] B.D. Lee, “Comparison of LDA and PCA for Korean Pro Go Player's Opening Recognition”, Journal of Korea Game Society, Vol. 13, No. 4, pp. 15-24, 2013.

[6] B.D. Lee and Y.W. Choi, “The best move sequence in playing Tic-Tac-Toe game”, Journal of The Korean Society for Computer Game, Vol. 27, No. 3, pp. 11-16, 2014.

[7] B.D. Lee, “Analysis of Korean, Chinese and Japanese Pro Go Player's Openings”, Journal of Korean Society for Computer Game, Vol. 26, No. 4, pp. 17-26, 2013.

[8] B.D. Lee, “Korean Pro Go Player's Opening Recognition Using PCA”, Journal of Korean Society for Computer Game, Vol. 26, No. 2, pp. 228-233, 2013.

[9] S. Gelly, M. Schoenauer, M. Sebag, O. Teytaud, L. Kocsis, D. Silver and C. Szepesvari, “The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions”, Communications of the ACM, Vol. 55, No. 3, pp. 106-113, 2012.

[10] S. Gelly and D. Silver, “Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go”, Artificial Intelligence, Vol. 75, Issue 11, pp. 1856-1875, 2011.

[11] Wikipedia, “Computer Go”, from http://en.wikipedia.org/wiki/Computer_Go, 2015.

[12] G. Chaslot, “Monte-Carlo Tree Search”, Ph.D. dissertation, University of Masstricht, 2010.

[13] Wikipedia, “Tic-Tac-Toe”, from http://en.wikipedia.org/wiki/Tic-Tac-Toe, 2015.

[14] A.A.J van der Kleij, “Monte Carlo Tree Search and Opponent Modeling through Player Clustering in no-limit Texas Hold'en Poker”, Master thesis, University of Groningen, 2010.

[15] H. Baier and M.H.M. Winands, “Monte-Carlo Tree Search and Minimax Hybrids”, Computer Games, Vol. 504, pp. 45-63, 2014.

[16] G. Hochmuth, “On the Genetic Evolution of a Perfect Tic-Tac-Toe Strategy”, from http://www.genetic-programming.org/sp2003/Hochmuth.pdf, 2015.

[17] N. Sephton, P.I. Cowling, E. Powley and N.H. Slaven, “Heuristic Move Pruning in Monte Carlo Tree Search for the Strategic

Card Game Lords of War", In Computational Intelligence and Games (CIG) of IEEE, pp. 1-7, 2014.
[18] T. Pepels, "Novel Selection Methods for Monte-Carlo Tree Search", Master thesis, University of Masstricht, 2014.
[19] D. Brand and S. Kroon, "Sample Evaluation for Action Selection in Monte Carlo Tree Search", from http://dl.acm.org/citation.cfm?doid=2664591.2664612, 2015.
[20] Y. Wang and S. Gelly, "Modification of UCT and sequence-like simulations for Monte-Carlo Go", from http://dept.stat.lsa.umich.edu/~yizwang/publications/wang07modifications.pdf, 2015.
[21] J.M. White, "Bandit Algorithms for Website Optimization", O'Relly, 2013.
[22] Ł. Lew, "Modeling Go Game as a Large Decomposable Decision Process", Ph.D. thesis, Warsaw University, 2011.
[23] P. Auer, N. Cesa-Bianchi and P. Fisher, "Finite-time Analysis of the Multiarmed Bandit Problem", Kluwer Academic Publishers, 2002.
[24] S. Takeuchi, T. Kanoke and K. Yamaguchi, "Evaluation of Monte Carlo Tree Search and the Application of Go", from http://www.csse.uwa.edu.au/cig08/Proceedings/papers/8046.pdf, 2015.
[25] I.J. Ahn and I.K. Park, "Design of Omok AI using Genetic Algorithm and Game Trees and Their Parallel Processing on the CPU", Journal of the Korea Information Science Society, Vol. 37, No. 2, pp. 66-75, 2010.
[26] A. Bhatt, P. Varshney and K. Deb, "In Search of No-loss Strategies for the Game of Tic-Tac-Toe using a Customized Genetic Algorithm", GECCO'08(Genetic and Evolutionary Computation Conference 2008, pp. 889-896, 2008.

이 병 두(Lee, Byung-Doo)

1982 한양대학교 원자력공학 학사
1991 서강대학교 정보처리학 석사
2005 Auckland University 컴퓨터공학 박사
2012~ 세한대 체육학부 바둑학과 조교수

관심분야 : 컴퓨터공학, 인공지능, 컴퓨터바둑

박 동 수(Dong-Soo Park)

현재 세한대학교 체육학부 부교수

관심분야 : 체육응용과학, 바둑게임

최 영 욱(Young-Wook Choi)

현재 세한대학교 체육학부 부교수

관심분야 : 스포츠사회학, 바둑게임