Journal of Korea Game Society 2018 Dec; 18(6): 59-68
http://dx.doi.org/10.7583/JKGS.2018.18.6.59

게임 프로그래밍

순수 몬테카를로 트리탐색을 기반으로 한 소형 바둑판에서의 가장 유망한 첫 수들

이병두
세한대학교 체육학부 바둑학과
blee026@korea.com

The most promising first moves on small Go boards, based on pure Monte-Carlo Tree Search

Byung-Doo Lee
Department of Baduk Studies, Division of Sports Science, Sehan University

요 약

간단한 규칙에도 불구하고 바둑은 인공지능 분야에서 가장 복잡한 전략적 보드게임 중의 하나이다. 몬테카를로 트리탐색(MCTS)은 최상우선 트리탐색 알고리즘으로 컴퓨터바둑 제작을 위해 사용되어 왔다. 저자는 9줄바둑판보다 작은 바둑판에서의 바둑게임 행위를 위해 MCTS를 활용하여 가장 유망한 첫 수를 찾고자 한다. 실험결과에 의하면 MCTS는 첫 수로 홀수형 바둑판에서는 정중앙, 짝수형 바둑판에서는 중앙 부근에 착수하기를 선호하는 것으로 나타났다.

ABSTRACT

In spite of its simple rule, Go is one of the most complex strategic board games in the field of Artificial Intelligence (AI). Monte-Carlo Tree Search (MCTS) is an algorithm with best-first tree search, and has used to implement computer Go. We try to find the most promising first move using MCTS for playing a Go game on a board of size smaller than 9×9 Go board. The experimental result reveals that MCTS prefers to place the first move at the center in case of odd-sized Go boards, and at the central in case of even-sized Go boards.

Keywords : small Go board(소형 바둑판), Monte-Carlo Tree Search(몬테카를로 트리탐색), promising first move(유망한 첫 수), odd- and even-sized Go boards(홀수형과 짝수형 바둑판들)

Received: Sep. 09. 2018 Revised: Nov. 22. 2018
Accepted: Nov. 22. 2018
Corresponding Author: Byung-Doo Lee(Sehan University)
E-mail: blee026@korea.com

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

바둑은 적어도 2,500년 전에 중국에서 기원된 2인-결정적-제로섬-완전정보게임이며, 그동안 컴퓨터가 인간을 제압 못하고 있었던 마지막 보드게임으로 포석, 중반전, 끝내기라는 세 단계로 진행이 된다[1]. 포석에서는 전략, 중반전에서는 전술, 마지막 단계인 끝내기에서는 선수, 후수, 집 크기 계산 등이 중요하다[2].

2015년 말에 Google DeepMind사의 AlphaGo-Fan이 최초로 19줄바둑에서 유럽 챔피언 Fan Hui를 5대0으로 제압하고, 이어서 2016년 초에 AlphaGo-Lee가 세계 최정상급 프로기사인 이세돌을 4대1로 승리하여 전 세계인들을 놀라게 했다. 또한 2016년 말부터 2017년 초에 걸쳐 AlphaGo-Master는 전 세계 프로기사들에게 60대0이라는 경이적인 승리 기록을 남겼으며, 이어서 동년에 Google DeepMind사는 인간 기보에 의존하지 않는 컴퓨터바둑의 최고 걸작인 AlphaGo-Zero를 남긴 후 바둑계를 은퇴했다. 이러한 역사적 대 사건으로 인해 인공지능은 전 세계인들에게 관심을 증폭시켰으며, 심지어 인공지능에 대한 경계심마저 일으키는 결과를 초래하였다[3].

컴퓨터바둑이 짧은 기간에 이렇게 경이적인 발전을 할 수 있었던 근원은 몬테카를로 트리탐색(MCTS)을 사용하였기 때문이다. 세계에서 가장 강한 컴퓨터바둑이 된 AlphaGo 역시 MCTS와 강화학습을 합성하여 제작되었다[4]. 본 논문에서는 순수 MCTS를 활용하여 소형 바둑판에서의 가장 유망한 첫 수와 제작 과정에서 발생되는 문제점을 찾고자 하였다.

2. 관련 연구

컴퓨터바둑은 전통적인 바둑을 둘 수 있는 컴퓨터 프로그램을 만드는 인공지능의 한 분야이다[5].

2.1 컴퓨터바둑의 등장

컴퓨터바둑은 1960년대 후반부터 시작되었으며, Zobrist는 1970년 그의 박사학위 논문에서 처음으로 완전한 컴퓨터바둑 프로그램을 구현해 냈다. 그 당시의 연구자들은 컴퓨터바둑의 기력향상을 시키는데 목적을 둔 것이 아니라, 바둑을 하나의 문제영역으로 삼아 연구자의 연구방법이나 새로운 기술을 바둑에서 시험하고자 했다. 특히 1980년대 중반에 급격한 개인용 컴퓨터의 보급과 잉창기 컵 대회의 상금 유혹으로 인해 상업용 컴퓨터바둑은 진일보하게 된다[6].

2.2 컴퓨터바둑의 진화

컴퓨터바둑의 진화 과정은 4세대로 분류할 수 있다. 제1세대는 MCTS가 나오기 이전의 휴리스틱(heuristic) 시대로 주로 1990년대부터 2000년대 초반이 된다. 이 시대의 컴퓨터바둑들은 주로 패턴인식과 인간기보 데이터베이스를 근간으로 한 휴리스틱 탐색을 구현해 냈다.

제2세대는 MCTS의 시대로 2000년대 중반부터 2015년 말 AlphaGo-Fan이 나오기 이전 시대가 된다. MCTS는 2008년에 Coulom에 의해 몬테카를로 방법을 게임트리에 접목하여 창안되어졌다[7]. 또한 MCTS를 이용한 컴퓨터바둑 Fuego가 2009년에 최초로 9줄바둑에서 최정상급 프로기사를 제압하기도 했다.

제3세대는 기계학습(ML: Machine Learning)을 위해 인공신경망(ANN: Artificial Neural Network)을 이용한 심층학습(deep learning)과 MCTS를 접목한 시대로 AlphaGo와 유사한 기법의 시대이다. 즉 엄청난 양의 인간 기보를 심층학습하고 MCTS를 활용한 시대로, 대표적인 컴퓨터바둑으로는 AlphaGo-Fan, AlphaGo-Lee와 AlphaGo-Master가 있다.

제4세대 컴퓨터바둑은 2017년에 등장한 AlphaGo-Zero와 같이 인간 기보의 도움없이 오로지 바둑규칙만을 이해한 후, 강화학습(RL: Reinforcement Learning)을 이용하여 자기학습(self-learning)을 한 후, 컴퓨터만

의 새로운 바둑세계를 발굴해낸 시대가 된다. 이와 비슷한 한 예는 1992년에 개발된 컴퓨터백가몬인 시차학습-가몬(TD-gammon)이 되며, 이는 인간이 시도도 못한 게임행위를 최초로 시도하여 새로운 백가몬게임 형태를 인간에게 보여주었다[8].

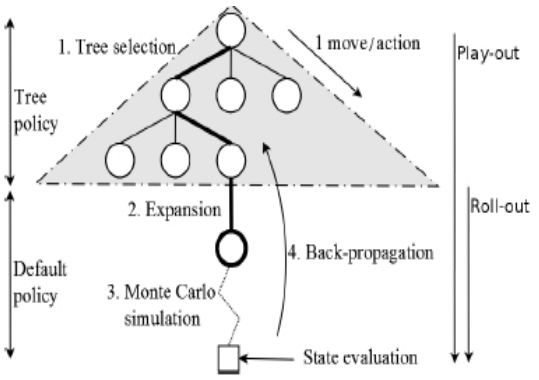
2.3 몬테카를로 트리탐색

게임프로그래밍에서는 최적의 트리탐색을 위해 평가함수를 활용한다[9]. 컴퓨터게임에서 최소최대탐색(min-max search) 또는 알파베타탐색($\alpha\beta$-search)을 사용하여 최적의 해를 구할 수 있으나, 컴퓨터바둑은 엄청난 게임깊이와 분기수로 인해 완전탐색을 수행할 수 없는 문제점이 있다. 결국 이러한 문제점의 대안으로 고안된 것이 MCTS가 된다. MCTS는 시뮬레이션을 통해 게임트리 내 하위트리를 임의로 표본추출을 한 후, 향후 시행할 행동들에 대한 값어치를 근사적으로 추정해 낸다[9].

MCTS는 바둑에서 국면평가함수(position evaluation function)를 활용하지 않는 최상우선탐색으로 탐색할 공간에 대해 임의적으로 탐험을 한다[3,10]. 이 방법은 점진적인 게임트리를 구축하기 위해 이전 탐험의 결과를 메모리에 저장하며, 지속적인 시뮬레이션을 통해 가장 유망한 차기 착점을 제시해준다. 결국 MCTS는 몬테카를로 시뮬레이션과 최소최대탐색을 접목해 놓은 것이다. 즉 MCTS는 임의 정책(randomized policy)에 의거하여 생성된 각 착점의 값어치를 평가하면서 게임트리를 구축 및 확장해 나간다[11].

MCTS는 탐색트리를 초기화한 후, [Fig. 1]에서와 같이 4단계의 작업인 선택(selection), 확장(expansion), 시뮬레이션(simulation), 역전파(backpropagation) 과정을 반복하여 진행한다[6,12-14].

(1) 선택단계: 반복적인 트리정책(tree policy)에 의해 게임트리 내 단말노드를 만날 때까지 자식노드를 계속 선택한다.
(2) 확장단계: 선택단계를 통해 찾아낸 자식노드가 탐색트리 내에 첨가되어 뿌리노드가 된다.
(3) 시뮬레이션단계: 시뮬레이션을 롤아웃(roll-out)이라고 하며, 탐색트리 내 뿌리노드로부터 디폴트정책(default policy)에 따라 시뮬레이션을 시작하며 단말노드를 만날 때까지 시뮬레이션을 지속한다.
(4) 역전파단계: 시뮬레이션단계로부터 얻어진 보상값을 갖고, 이 값을 선택단계에서 선발된 자식노드로부터 게임트리 내에 있는 뿌리노드로 역전파해가며 그 값을 갱신해 나간다.



[Fig. 1] The process of MCTS [14]

2.4 소형 바둑 관련연구

대부분의 컴퓨터바둑 연구는 19줄바둑을 중심으로 이루어지고 있으나, 2003년에 Werf 등은 알파베타탐색을 사용하여 [Table 1] 왼쪽에서 보듯이 5줄 이하 규모의 정사각형 바둑에 대해 분석을 하여 게임을 위한 첫 수들의 위치가 중앙 부근이 됨을 보였으며, 아울러 일련의 수순처리를 위한 게임트리의 규모를 보였다[15]. 이어서 2009년에 Werf와 Winands는 1×2줄바둑에서부터 3×10줄바둑까지의 직사각형 바둑에 대해 알파베타탐색을 활용하여 바람직한 첫 수 및 일련의 수순을 보이려고 했다[16]. 한편 Lee는 순수 MCTS를 활용하여 9줄바둑에서의 가장 바람직한 첫 수는 정중앙(천원)이 됨을 보였다[1].

[Table 1] First move and elapsed time

Board size	$\alpha\beta$-search (1st move)	pure MCTS	
		1st move	Elapsed time
2×2	any	any	9s(0.15m)
3×3	center	center	10s(0.17m)
4×4	central	central	166s(2.77m)
5×5	center	center	598s(9.97m)
6×6	-	central	1581s(26.35m)
7×7	-	center	3278s(54.63m)
8×8	-	central	5165s(86.08m)
9×9	-	center	6465s(107.75m)

3. 실험방법 및 결과

3.1 실험방법

19줄바둑은 전 세계에서 공식 대국용으로 사용되고 있으며, 9줄바둑은 주로 바둑 입문자나 어린이 학습을 위해 사용되고 있다[17]. 본 논문에서는 9줄 이하의 규모를 소형 바둑으로 간주하였으며, 기존 논문과는 달리 소형 바둑 전체에 대한 가장 바람직한 첫 수 및 이에 대응하는 두 번째 수인 백의 착점 위치를 알아보려고 했다.

본 실험에서는 아무런 전략을 구사하지 않는 순수 MCTS를 활용하였으며, 소형 바둑에서 가장 바람직한 첫 수를 찾기 위해 1년여에 걸쳐 Window 8.1 64bit 운영체제에서 Intel Core i5-3337U 프로세서를 이용한 C++ 프로그래밍을 구현하였다. 또한 순수 MCTS 구현을 위한 소스 코드를 작성 시에 고려된 사항으로는 게임 중 두 대국자가 연속으로 순서 넘김 (pass)을 한 경우에 대국을 종료하였으며, 게임이 종료된 후 대국의 결과를 위해 중국 식 규칙을 적용한 계가를 적용하였으며, 이때 덤(Komi)은 고려하지 않았다.

소형 바둑에서의 가장 유망한 첫 수의 위치를 알아내기 위해 착수 가능한 모든 위치에 대해 수많은 롤아웃을 실시하여 [Fig. 2]와 같은 결과를 얻을 수 있었다. 참고로 한 번의 롤아웃 후의 보상값(또는 결과값)인 R을 위해 식 (1)과 같이 먼저 두는 대국자인 흑이 이기는 경우에는 +1, 나중에 두는 대국자인 백이 이기는 경우에는 0, 그리고 무승부인 경우에는 +0.5를 부여하였다.

$$R = \begin{cases} 1, & \text{if black won} \\ 0.5, & \text{if both tied} \\ 0, & \text{if White won} \end{cases} \quad (1)$$

또한 s_t인 상태에서 착수 가능한 임의의 위치인 p에 대해 롤아웃을 실시한 후의 보상값인 $Q(s_t,a_p)$는 식 (2)와 같이 계산되어진다.

$$Q(s_t,a_p) = \frac{1}{N(s_t,a_p)} \sum_{i=1}^{N(s_t)} \Pi_i(s_t,a_p) R_i \quad (2)$$

여기서 $N(s_t,a_p)$는 현재상태 s_t로부터 선택된 착수행위 a_p의 롤아웃의 수가 되며, $N(s_t)$는 상태 s_t를 통해 수행된 전체 롤아웃의 수가 된다. R_i는 s_t로부터 수행된 i번째 롤아웃의 보상값이 되며, $\Pi_i(s_t,a_p)$는 i번째 롤아웃에 대해 상태 s_t로부터 행위 a_p가 선택된 경우에는 1이 되며, 그 이외의 경우에는 0인 된다.

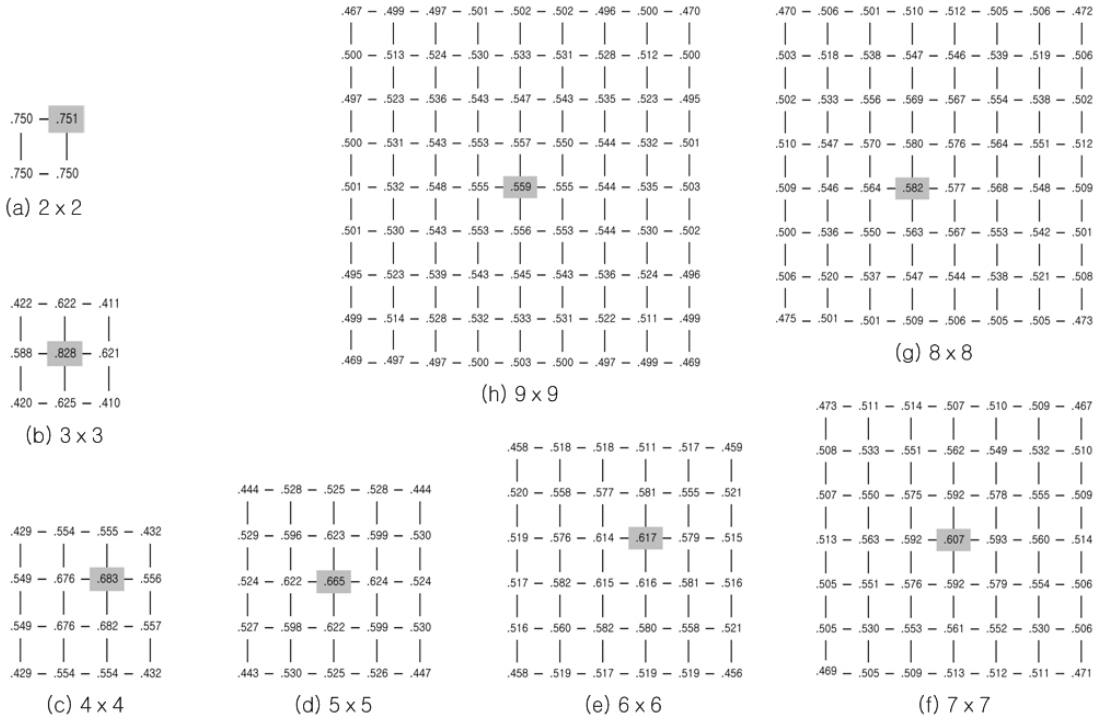
롤아웃 이후 대국자의 최종 착수 a_t는 식 (3)에서와 같이 최대 보상값을 갖는 착점 위치인 p가 된다.

$$a_t = \text{argmax}_{a_p \in A(s_t)} Q(s_t,a_p) \quad (3)$$

여기서 $A(s_t)$는 상태 s_t에서 착수 가능한 모든 행위 a_p가 된다.

3.2 실험결과

실험 결과에 따르면 [Table 2]에서 보듯이 $n \times n$인 정사각형 바둑판에 대해 n이 짝수인 경우, 즉 $n = 2i (i = 1,2,3,4)$인 바둑판인 경우에는 천원



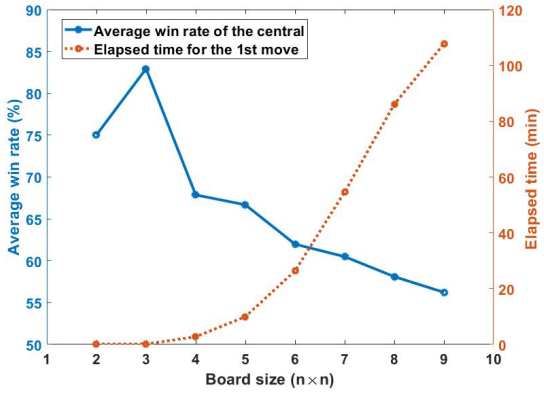
[Fig. 2] Win rates of the first moves

주변이 가장 평균승률이 높으며, 반면에 n이 홀수인 경우, 즉 $n=2i+1(i=1,2,3,4)$인 바둑판인 경우에는 천원이 가장 평균승률이 높은 것으로 나타났다.

[Table 2] Average win rates of the central

	$i=1$	$i=2$	$i=3$	$i=4$
$n=2i$	75.03± 0.02%	67.94± 0.18%	61.79± 0.04%	57.87± 0.07%
$n=2i+1$	82.81%	66.53%	60.67%	55.93%

결국 소형 바둑판에서의 첫 수로 [Table 1]의 오른쪽과 같이 흑이 천원(center) 또는 천원 주변(central)을 흑1로 두게 되면 [Table 2]와 [Fig. 3]에서 보듯이 평균승률이 50%를 훨씬 상회하기 때문에 게임에서 쉽게 지지 않음을 알 수 있다.



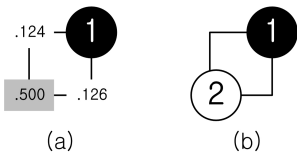
[Fig. 3] Average win rate of the central Vs Elapsed time for the first move (board size adjusted)

[Fig. 3]을 살펴보면 바둑판의 규모가 점차 커짐에 따라 평균승률이 83%정도에서 56%정도로 급속히 감소함을 알 수 있다. 이는 바둑판의 규모가 커짐에 따라 중앙 부근에 대한 첫 수로서의 착점

값어치가 급격히 떨어짐을 의미한다. 또한 시뮬레이션을 통해 MCTS가 첫 수를 구해내는 컴퓨팅 시간을 살펴보면 [Table 1]과 [Fig. 3]에서 보듯이 2줄바둑에서는 9초, 9줄바둑에서는 108분이 되어 바둑판의 규모가 커짐에 따라 컴퓨팅 시간이 기하급수적으로 증가함을 알 수 있다. 한편 2줄바둑에서 9줄바둑까지의 세부 실험결과는 다음과 같다.

3.2.1 2줄바둑에서의 실험결과

네 귀로만 구성된 된 2줄바둑에서 첫 번째 대국자인 흑1의 최상의 첫 수는 [Fig. 2](a)에서 보듯이 착수 가능한 4곳의 평균승률이 75.03±0.02%가 되어, 아무 곳에 두어도 쉽게 게임에 지지 않는다. 또한 두 번째 대국자인 백의 최상의 다음 수는 [Fig. 4](a)와 같이 흑1의 마주 보는 곳의 평균승률이 50.0%가 되어, 백2로 [Fig. 4](b)와 같이 두어 빅이 되기 때문에 게임은 무승부가 된다.



[Fig. 4] Sequence of moves

참고로 몬테카를로 시뮬레이션 횟수를 크게 하면 그 추정값은 점점 정확하게 된다. 즉 시뮬레이션 횟수 N이 증가할수록 그 추정값은 모집단의 평균에 근접하게 된다. [Table 3]은 2줄바둑에서 시뮬레이션 횟수에 따른 첫 수를 구하기 위해 소모된 컴퓨팅 시간, 4곳에 대한 평균승률, 표준편차, 표준오차가 된다.

[Table 3] A comparison of statistic

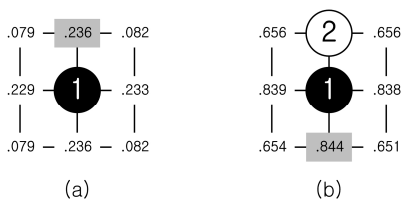
		$N=10^3$	$N=10^4$	$N=10^5$	$N=2.5\times10^5$	$N=10^6$
time		0.05s	0.36s	3.70s	9.08s	35.98s
pos	p1	75.85%	75.34%	74.94%	75.01%	74.98%
	p2	74.80%	74.89%	75.04%	74.97%	75.04%
	p3	74.05%	75.21%	75.12%	75.05%	75.00%
	p4	76.40%	74.83%	75.01%	75.03%	75.03%
mean		75.28%	75.07%	75.03%	75.02%	75.01%
SD		1.05%	0.25%	0.07%	0.03%	0.03%
SE		0.53%	0.12%	0.04%	0.02%	0.01%

한 예로 [Table 3]을 살펴보면 시뮬레이션의 수가 $N=10^5$인 경우의 착수 가능한 4곳의 평균승률은 각각 74.94%(p1), 75.04%(p2), 75.12%(p3), 75.01%(p4)로 나타났으며, 이에 대한 평균승률은 75.03%, 표준편차는 0.07%, 표준오차는 0.04%, 즉 75.03±0.04%를 보였다.

본 실험에서는 시뮬레이션을 통해 구해지는 보상값의 정확성과 컴퓨팅 시간을 고려하여 임의로 $N=2.5\times10^5$번의 시뮬레이션을 적용하였다.

3.2.2 3줄바둑에서의 실험결과

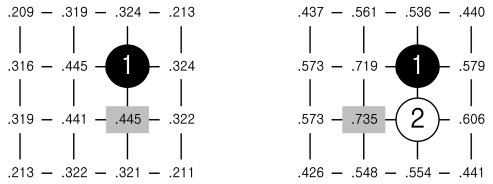
3줄바둑은 중앙, 변, 귀로 형성된 소형 바둑이다. [Fig. 2](b)에서 보듯이 중앙, 변, 귀의 평균승률이 각각 82.81%, 61.39±0.87%, 41.58±0.30%가 되어, 첫 수인 흑1로 평균승률이 가장 높은 천원에 두게 된다. 이후 백은 [Fig. 5](a)에서 보듯이 23.6%의 평균승률을 보인 상변에 백2로 두게 된다. 참고로 흑3은 [Fig. 5](b)에서 보듯이 평균승률 84.4%를 보인 하변에 두게 된다.



[Fig. 5] Sequence of moves

3.2.3 4줄바둑에서의 실험결과

4줄바둑은 정중앙, 즉 천원이 없는 형태의 소형 바둑이다. [Fig. 2](c)에서 보듯이 천원 주변의 평균승률이 67.94±0.18%가 되어, [Fig. 6](a)에서 보듯이 흑은 첫 수로 천원 주변에 흑1로 두게 된다.

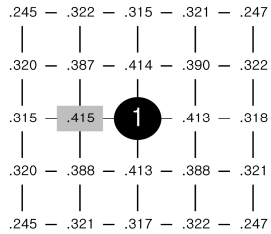


[Fig. 6] Sequence of moves

이후 백은 [Fig. 6](a)에서 보듯이 44.5%의 평균승률을 보인 곳을 [Fig. 6](b)에서와 같이 백2로 두어 흑과의 전투를 벌이게 된다. 이후 흑의 향후 착수 위치를 살펴보면 [Fig. 6](b)에서 보듯이 평균승률 73.5%를 보인 곳을 흑3으로 두어 백2를 제치게 된다.

3.2.4 5줄바둑에서의 실험결과

5줄바둑에서의 평균승률을 살펴보면 [Fig. 2](d)에서 보듯이 정중앙이 66.53%가 되어, [Fig. 7]과 같이 흑1로 천원에 두게 된다.

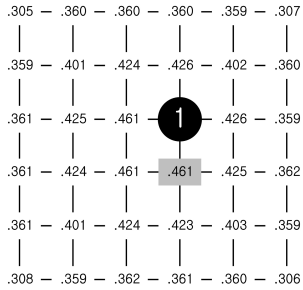


[Fig. 7] Win rates of each position on a 5×5 Go board

흑1 이후 백2의 착점을 구하기 위해 각 점에 대한 평균승률을 살펴보면 흑1의 곳인 천원과 바로 인접한 4곳의 평균승률이 41.37±0.03%가 되며, 백2로는 이 중 가장 큰 값(41.5%)을 취해, 즉 흑1의 왼쪽을 붙여 전투를 벌이게 된다.

3.2.5 6줄바둑에서의 실험결과

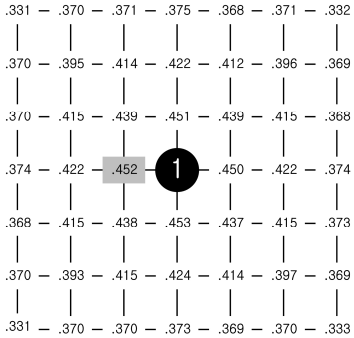
6줄바둑에서의 평균승률을 살펴보면 [Fig. 2](e)에서 보듯이 천원 주변의 평균승률이 61.79±0.04%가 되어, [Fig. 8]에서 보듯이 흑은 첫 수인 흑1로 천원 주변을 두게 된다. 이후 백은 [Fig. 8]에서 보듯이 평균승률 46.1%를 보인 곳에 백2로 두어, 즉 흑1 밑에 붙여 전투를 벌이게 된다.



[Fig. 8] Win rates of each position on a 6×6 Go board

3.2.6 7줄바둑에서의 실험결과

7줄바둑에서의 평균승률을 살펴보면 [Fig. 2](f)에서 보듯이 천원이 60.67%가 되어, [Fig. 9]와 같이 흑1로 천원에 두게 된다.



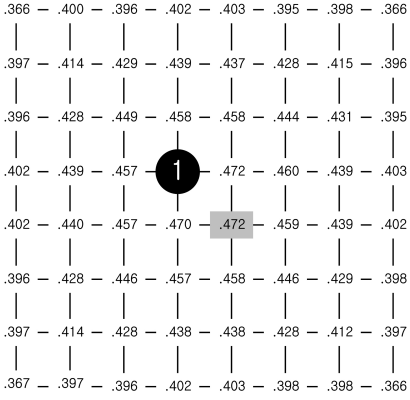
[Fig. 9] Win rates of each position on a 7×7 Go board

이후 [Fig. 9]에서 보듯이 백의 다음 착점을 위

해 각 점에 대한 평균승률을 살펴보면 흑1의 곳인 천원과 바로 인접한 4곳의 평균승률이 45.19±0.07%가 되며, 백2로는 이 중 가장 큰 값(45.2%)을 취해, 즉 흑1의 왼쪽에 붙여 전투를 벌이게 된다.

3.2.7 8줄바둑에서의 실험결과

8줄바둑에서의 평균승률을 살펴보면 [Fig. 2](g)에서 보듯이 천원 주변의 평균승률이 57.87±0.07%가 되어, [Fig. 10]에서 보듯이 흑은 첫 수인 흑1로 천원 주변을 두게 된다.

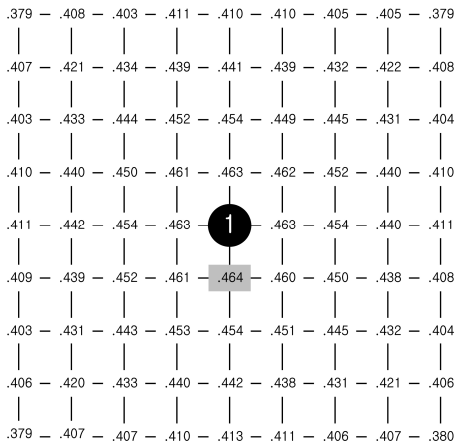


[Fig. 10] Win rates of each position on a 8×8 Go board

이후 백은 [Fig. 10]에서 보듯이 평균승률이 47.2%로 가장 높은 곳, 즉 흑1을 마주보는 마름모인 곳에 백2로 두어 전투 준비를 하게 된다.

3.2.8 9줄바둑에서의 실험결과

공식 대국용으로 사용되는 19줄바둑의 축소판인 9줄바둑은 주로 바둑 입문자나 어린이 교습용으로 사용되고 있다. [Fig. 2](h)에서 보듯이 천원의 평균승률이 55.93%가 되어, 흑은 첫 수인 흑1로 이 곳을 두게 된다. 이후 백2는 [Fig. 11]에서 보듯이 평균승률이 46.4%로 가장 높은 곳, 즉 흑1의 밑에 붙여 곧바로 흑과의 전투를 벌이게 된다.



[Fig. 11] Win rates of each position on a 9×9 Go board

4. 결 론

본 연구는 인공지능을 이용하여 19줄바둑판에서의 컴퓨터바둑에 대한 기력을 향상시키는 기존의 기술개발과는 달리, 순수 MCTS를 활용하여 9줄이하의 소형 바둑에서의 가장 바람직한 첫 수를 찾고자 했다. 실험을 통해 알게 된 사실은 다음과 같다.

홀수형 소형 바둑판에서의 가장 바람직한 첫 수는 정중앙이 되며, 짝수형 소형 바둑판에서의 가장 바람직한 첫 수는 중앙주변이 되었다. 또한 실험을 통해 밝혀진 또 다른 제작상의 문제점은 유망한 착점을 찾아내기 위한 컴퓨팅 시간이 너무 많이 걸린다는 사실이다. 이를 극복하기 위해서는 MCTS 내의 트리정책을 개선하여 선택단계에서 유망한 착점을 짧은 시간 내에 집중적으로 선택하는 알고리즘이 필요하며, 아울러 디폴트정책을 개선하여 게임의 종료를 알리는 단말노드까지의 접근시간을 대폭 단축할 수 있는 알고리즘의 개발이 절실함을 알 수 있었다.

향후 본 논문의 활용방안과 기대효과를 살펴보면, 활용방안에 있어서 본 논문의 접근방법과 실험결과를 바탕으로 모든 유사 보드게임 및 관련 게임에 이를 확대 적용할 수 있으며, 기대효과 측면에서 보면 타 문제영역에서의 유사 실험을 위한

기초자료 제공 및 실제 제작 시 발생될 문제점을 사전 제시해준 효과가 있다.

REFERENCES

[1] B.D. Lee, “The first move in the game of 9×9 Go, using non-strategic Monte-Carlo Tree Search” Journal of Korea Game Society, Vol. 17, No. 3, pp.63-70, 2017

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

[3] B.D. Lee and Y.W. Choi, “Monte-Carlo Game Implementation”, Hansan Press, 2017

[4] B.D. Lee, “Competition between MCTS and UCT in the game of Tic-Tac-Toe” Journal of Korean Society for Computer Game, Vol. 29, No. 1, pp.1-6, 2016

[5] Wikipedia, “Computer Go”, https://en.wikipedia.org/wiki/Computer_Go, October, 2018

[6] B.D. Lee, “AI Go”, Books Holic, 2011

[7] R. Coulom, “The Monte-Carlo Revolution in Go”, Japanese-French Frontiers of Science Symposium, 2009

[8] B.D. Lee, “Monte-Carlo Tic-Tac-Toe”, Hansan Press, 2015

[9] B.D. Lee, D.S. Park, and Y.W. Choi, “The UCT algorithm applied to find the best first move in the game of Tic-Tac-Toe” Journal of Korea Game Society, Vol. 15, No. 5, pp.109-118, 2015

[10] M.H.M. Winands and Björnsson, “Evaluation Function Based Monte-Carlo LOA”, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.2046&rep=rep1&type=pdf, October, 2018

[11] 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

[12] B.D. Lee, “Enhanced strategic Monte-Carlo Tree Search algorithm to play the game of Tic-Tac-Toe” Journal of Korea Game Society, Vol. 16, No. 4, pp.79-86, 2016

[13] T. Pepels, “Novel Selection Methods for Monte-Carlo Tree Search”, Master thesis, Maastricht University, 2014

[14] D. Perez, S. Samothrakis, and S. M. Lucas. “Knowledge-based fast evolutionary MCTS for general video game playing”. IEEE Conference on Computational Intelligence and Games (CIG), pp.1-8, 2014

[15] E.C.D. Werf, H.J. Herik, and J.W.H.M Uiterwijk, “Solving Go on Small Boards”, ICGA Journa, Vol. 26, No. 2, pp.92-107, 2003

[16] E.C.D. Werf and M.H.M. Winands, “Solving Go for Rectangular Boards”, ICGA Journa, Vol. 26, No. 2, pp.92-107, 2009

[17] Sense′s Library, “Small board Go”, from https://senseis.xmp.net/?SmallBoardGo, 2018.

이 병 두 (Lee, Byung-Doo)

약 력 : 1982 한양대학교 원자력공학 학사
1991 서강대학교 정보처리학 석사
2005 Auckland University 컴퓨터공학 박사
2010- 세한대학교 체육학부 바둑학과 조교수

관심분야 : 컴퓨터공학, 인공지능, 컴퓨터바둑