Journal of Korea Game Society 2016 Aug; 16(4): 79-86
http://dx.doi.org/10.7583/JKGS.2016.16.4.79

게임 프로그래밍

삼목 게임을 위해 개선된 몬테카를로 트리탐색 알고리즘

이병두
세한대학교 체육학부 바둑학과
blee026@korea.com

Enhanced strategic Monte-Carlo Tree Search algorithm to play the game of Tic-Tac-Toe

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

요 약

몬테카를로 트리탐색은 최대우선탐색 알고리즘이며, 많은 게임 특히 바둑 게임에 성공적으로 적용되어 왔다. 삼목 게임에서 MCTS 간의 대국을 통해 성능을 평가하고자 했다. 첫 번째 대국자는 항상 두 번째 대국자에 비해 압도적인 우위를 보였으며, 최선의 게임 결과가 무승부가 됨에도 불구하고 첫 번째 대국자가 두 번째 대국자에 비해 우월한 이유를 찾고자 했다. MCTS는 반복적인 무작위 샘플링을 기반으로 하는 통계적 알고리즘이기 때문에, 특히 두 번째 대국자를 위해 전략을 요하는 시급한 문제를 적절히 대처하지 못한다. 이를 위해 전략적 MCTS(S-MCTS)를 제안하며, S-MCTS는 결코 삼목 게임에서 지지 않는다는 것을 보였다.

ABSTRACT

Monte-Carlo Tree Search(MCTS) is a best-first tree search algorithm and has been successfully applied to various games, especially to the game of Go. We evaluate the performance of MCTS playing against each other in the game of Tic-Tac-Toe. It reveals that the first player always has an overwhelming advantage to the second player; and we try to find out the reason why the first player is superior to the second player in spite of the fact that the best game result should be a draw. Since MCTS is a statistical algorithm based on the repeated random sampling, it cannot adequately tackle an urgent problem that needs a strategy, especially for the second player. For this, we propose a strategic MCTS(S-MCTS) and show that the S-MCTS player never loses a Tic-Tac-Toe game.

Keywords : Go(바둑), Tic-Tac-Toe(삼목), MCTS(몬테카를로 트리탐색), S-MCTS(전략적 몬테카를로 트리탐색), Monte-Carlo Tree Search(몬테카를로 트리탐색)

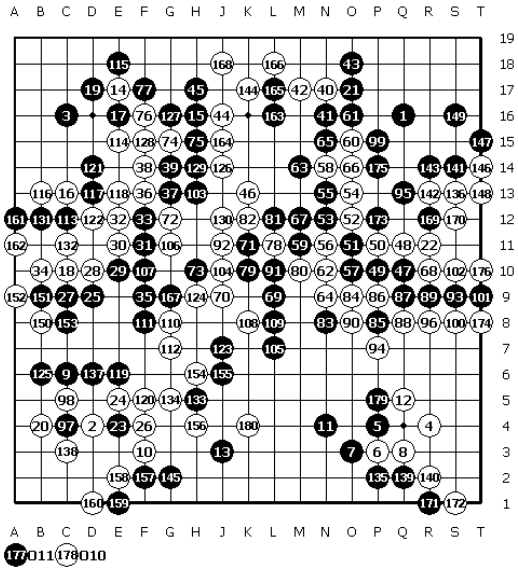
Received: May. 9, 2016 Revised : Jul, 4, 2016
Accepted: Jul, 7, .2016
Corresponding Author: Byung-Doo Lee (Sehan University)
E-mail: blee026@korea.com

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

적어도 2,500년 전 중국에서 기원된 바둑은 아직까지 컴퓨터가 인간을 완전 제압을 못하고 있었던 마지막 보드게임이었다[1,2,3,4].

올해 1월 27일 <네이처>지에 따르면 "구글 DeepMind의 AlphaGo가 2015년 10월에 유럽 바둑챔피언 Fan Hui 프로 2단을 5:0으로 제압했고, 여타 컴퓨터 바둑과의 대국에서도 99.8%(495전 494승)의 압승을 거두었다."고 발표했다[5]. 또한 AlphaGo는 금년 3월 9일~15일에 걸쳐 벌어진 세계 최강의 바둑 프로기사인 이세돌 9단과의 공식 5번기 대국([Fig. 1] 참조)에서 4:1로 승리를 거두는 쾌거를 이루어 냈다. 이는 인공지능 분야에서 경이적인 결과로 해석되며, 1차(1784년: 기계적 생산, 증기기관), 2차(1870년: 대량생산, 전기에너지), 3차(1969년: 전자장치, 정보기술)에 이어 인공지능이 4차 산업혁명(현재: 인공지능, 빅데이터)을 견인하는 주요 역할을 담당할 것을 보여 주었다[6].



[Fig. 1] The fourth game record, with Lee Sedol's win by resignation, played between AlphaGo(Black) and Lee Sedol(white)

1990년대부터 인공지능 연구자들은 최신 기법을 적용하여 컴퓨터 바둑의 기력을 향상시키고자 했으나 그 기력이 프로기사에 못 미쳐 새로운 방법을 고대하고 있었다[1,2,3,7].

그 대안 책으로 몬테카를로 트리탐색(MCTS: Monte-Carlo Tree Search)을 적용한 결과 마침내 2007년에 MoGo가 9줄바둑에서 Guo Juan 프로 5단을 제압했으며, 이어서 2009년에는 Fuego가 최정상 기사인 Zhou Junxun 프로 9단을 누르면서 9줄바둑에서 컴퓨터가 프로기사를 완전히 제압했다[1,7,8,9]

한편 [Table 1]에서 보듯이 19줄바둑에서도 선전을 하여 2008년에 MoGo는 Kim Myungwan 프로 8단을 9점 접바둑으로 이겼으며, 2015년에 AlphaGo는 Fan Hui 프로 2단을 맞바둑으로 승리를 거둠에 이어, 올해에는 최정상 기사인 Lee Sedol 프로 9단을 제압함으로서 마침내 컴퓨터 바둑이 프로기사보다 한 수 위라는 사실을 입증했다.

[Table 1] The first computer Go programs won against professional Go players[1,2,5,7,8,10,11]

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
2015	0	2 dan	AlphaGo
2016	0	9 dan	AlphaGo

이렇게 컴퓨터 바둑이 단기간에 비약적인 발전을 할 수 있는 원동력은 MCTS의 활용에 있었다. 세계 최강의 AlphaGo 역시 통제학습으로 훈련된 심층신경망(DNN: Deep Neural Network)을 이용한 MCTS, 그리고 자기 자신과의 게임을 통한 강화학습을 합성한 기법으로 이 역시 MCTS를 적용

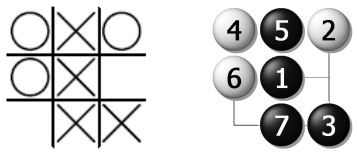
하였기에 이러한 놀라운 성과가 가능하였다[2,5].

인공지능 분야에서 이러한 위대한 역사를 만들어 낸 MCTS의 적극적인 활용을 위해, 저자는 소규모의 게임인 삼목 게임을 통해 순수 MCTS의 문제점과 이에 대한 대처 방안을 제시하고자 한다.

2. 개요 및 관련 연구

2.1 삼목 게임

삼목 게임(Tic-Tac-Toe)은 전 세계적으로 잘 알려진 게임 중의 하나로 두 대국자가 3×3칸으로 된 종이 위에 X와 ○를 번갈아 연필로 써서 가로, 세로 또는 대각선상에 동일한 모양이 연속하여 3개가 형성되면 이기는 게임이다[12]. 참고로 [Fig. 2](a)는 첫 번째 대국자인 X가 세로 방향으로 승리를 한 장면이다. 또한 저자는 삼목 게임에서 일련의 수순을 표시하기 위해 [Fig. 2](a)와 같은 X와 ○의 형태가 아닌 [Fig. 2](b)와 같이 흑돌과 백돌 위에 수순을 표시하고자 했다.

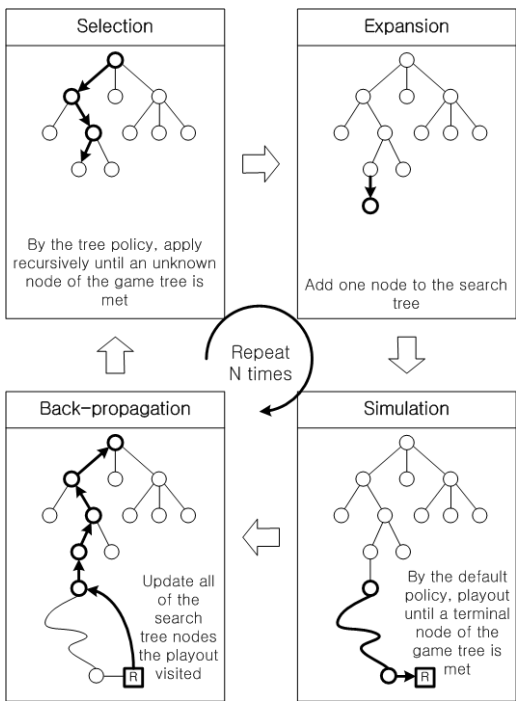


[Fig. 2] (a) A finished Tic-Tac-Toe game won by the first player, X: (b) A move sequence of the game in the form of (a)

2.2 몬테카를로 트리탐색

대부분의 게임프로그래밍은 평가함수(evaluation function)를 이용한 게임트리탐색를 활용한다[1,2]. 게임트리탐색에서의 최적의 해를 구하는 데에 있어 고려해야 할 사항으로는 게임트리의 분기수와 깊이의 규모에 있다. 분기수와 깊이가 비교적 작은 경우에는 최소최대탐색 또는 알파-베타 가지치기를 수행하여 최적의 해를 구할 수 있으나, 바둑과 같이 분기수와 깊이가 매우 큰 경우에는 이를 이용한 전역탐색(exhaustive search)을 수행할 수가 없어 MCTS를 활용해야 한다[1,2,13].

MCTS 알고리즘을 수행하기 위해서는 (1)게임 중 발생되는 행위들에 대한 정보를 담는 게임트리(game tree)와 (2)MC 시뮬레이션을 위해 사용되는 정보를 담는 탐색트리(search tree)가 필요하며 [1,2,13], [Fig. 3]에서 보듯이 선택, 확장, 시뮬레이션, 역전파라는 4단계를 거친다[1,2,7,14,15].



[Fig. 3] Four steps for conducting the MCTS algorithm[1,2,7,16,17]

특히 제3단계인 시뮬레이션 단계에서는 MC 시뮬레이션을 수행하게 되며, 수행 후 현재의 국면에서의 각 자식노드에 대한 평가값은 MC 시뮬레이션을 통해 얻은 결과값의 평균을 적용할 수 있다. 이후 각 자식노드들의 평균치로부터 최댓값을 갖는 자식노드를 선택하여 일련의 게임 행위를 수행하게 된다. 이를 위한 유사코드는 [Fig. 4]와 같다.

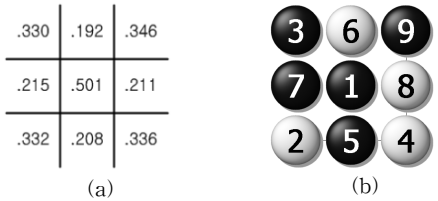
```
function MCTS(Position, NoOfSimulations)
  for each child n available from Position
    reward[n] = 0
    for i from 1 to NoOfSimulations
      POS = copy(Position)
      while (POS is terminal node)
        MCTS(POS, random child from POS)
      end while
      reward[n] += result(POS)
    end for
    reward[n] /= NoOfSimulations
  end for
  return(child i with highest reward[i])
end function
```

[Fig. 4] Pseudo-code for the MCTS algorithm[2]

2.3 기 수행된 관련 연구

저자는 본격적인 MCTS 활용을 위해 이를 삼목 게임에 적용한 결과 다음과 같은 사실을 도출해 냈다.

■ 삼목 게임에서 첫 수로 착수 가능한 9곳 중 승률이 가장 높은 첫 수 위치를 찾아내기 위해 100,000번의 MC 시뮬레이션을 수행한 결과, [Fig. 5](a)에서 보듯이 첫 수로 중앙에 둔 경우 50.1%, 귀에 둔 경우 평균 33.6%, 변에 둔 경우 평균 20.6%의 승률을 보여, 첫 수로 “중앙에 두는 것이 가장 좋고, 변에 두는 것이 가장 나쁘다.”라는 사실을 최초로 밝혀냈다[1,4,7].



(a) (b)

[Fig. 5] (a) Win rates of each position after 100,000 MC simulations for a game and (b) A best move sequence for playing a game[3,7]

■ 또한 삼목 게임에서 인간과 컴퓨터 간의 대국을 통해 두 대국자의 최선의 수순과 게임 결과를 알아보기 위해 MCTS를 적용한 결과 [Fig. 5](b)와 같은 최선의 수순의 한 예를 보였으며, 아울러 인간과 컴퓨터 간의 최선의 게임결과는 무승부가 됨을 보였다[1,7,8].

3. 실 험

3.1 MCTS 간의 대국

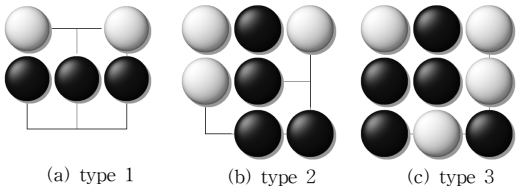
우선 MCTS의 성능을 측정하기 위해 MCTS 간에 10,000번의 삼목 게임을 수행했으며, 한 게임 당 1개의 최선의 착수를 결정짓기 위해 100,000번의 MC 시뮬레이션을 시켰다. 게임의 결과는 이긴 경우 +1점, 비긴 경우 +0.5점, 진 경우는 0점으로 처리하였다.

[Table 2]에서 보듯이 MCTS 간의 10,000번의 게임 결과 첫 번째 대국자(MCTS)가 두 번째 대국자(MCTS)에 대해 평균 승률이 95% 신뢰도 범위 내에서 80.3±0.8%를 보였다. 이 결과로 먼저 두는 첫 번째 대국자(MCTS)가 두 번째 대국자(MCTS)에 비해 압도적으로 유리함을 알 수 있었다. 참고로 첫 번째 대국자(MCTS)는 10,000번의 게임 중 6,069번의 승리와 3,931번의 무승부를 기록하였다.

[Table 2] Win rates of the first player playing against the second player in 10,000 games

1st player \ 2nd player	MCTS	S-MCTS
MCTS	80.3±0.8%	50.0±1.0%
S-MCTS	80.3±0.8%	50.0±1.0%

한편 [Fig. 6]에서 보듯이 첫 번째 대국자(MCTS)인 흑이 승리와 무승부를 거둔 전체 유형을 살펴보면, 유형 1, 2와 같이 승리를 하는 경우와 유형 3과 같이 무승부를 기록한 유형으로 나누어짐을 알 수 있었다.

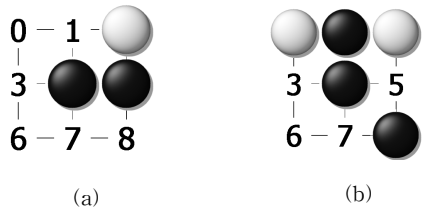


[Fig. 6] Three patterns in Black's game results

앞서 설명했듯이 두 대국자간의 최선의 게임 결과는 무승부이어야 하나, 첫 번째 대국자(MCTS)인 흑이 승리를 한 이유는 두 번째 대국자(MCTS)인 백이 최선의 대응을 못했기 때문이다. 이러한 현상은 MC 시뮬레이션을 수행 시 발생되는 통계적 확률값에서 그 이유를 찾을 수 있었다.

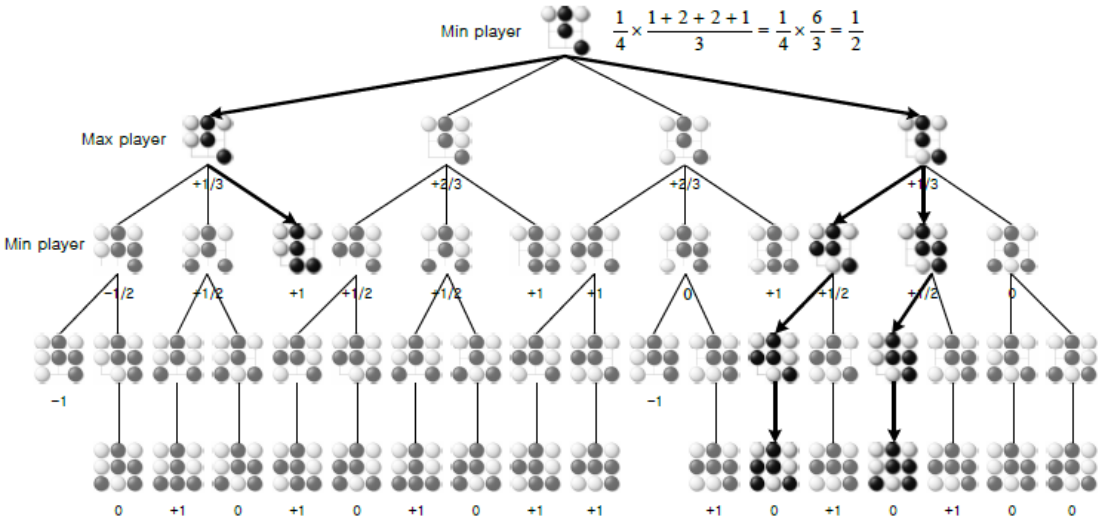
3.2 수학적 확률값과 통계적 확률값

MCTS는 반복적인 무작위적 탐험을 통해 얻어낸 (수학적 확률값의 근사값인) 통계적 확률값으로 최적의 해를 구해내는 것이다[18]. [Fig. 7]은 MCTS 간의 대국에서 벌어진 실제 상황이다. [Fig. 7](a)와 같은 상황에서 두 번째 대국자인 백(최소 대국자)은 3의 곳을 반드시 둔다고 장담할 수 있을까?



[Fig. 7] Two patterns happened in games[18]

실제 게임트리를 그려 0의 곳과 3의 곳에 대한 수학적 확률값을 계산해 보면 0의 곳은 $\frac{1}{5}\times\frac{1}{4}\times\frac{16}{3}$이 되며, 3의 곳은 $\frac{1}{5}\times\frac{1}{4}\times\frac{22}{3}$가 되어 최소 대국자인 백은 다음 착수로 3의 곳이 아닌 최솟값을 갖는 0의 곳을 선택하게 된다. 또한 [Fig. 7](b)와 같은 상황에서 백은 7의 곳을 반드시 둔다고 장담할 수 있을까? 이 역시 [Fig. 8]과 같이 게임트리를 그려 3의 곳과 7의 곳에 대한 수학적 확률값을 계산해 보면 3의 곳은 $\frac{1}{3}$, 7의 곳 역시 $\frac{1}{3}$로 같게 되어 백은 다음 착수로 반드시 7의 곳을 선택한다는 보장이 없다.



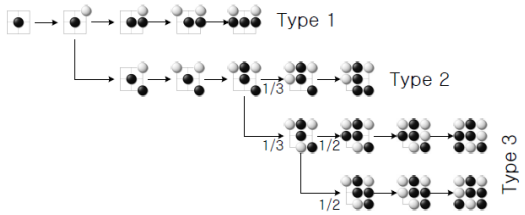
[Fig. 8] Minimax search tree for playing the game with board position in [Fig. 7](b)

결국 반복적인 무작위 탐험에 의해 계산된 통계적 확률값에 전적으로 의존하는 MCTS는

■ 수학적 확률값이 같은 경우 우리가 원하는 곳에 착수를 안 할 수도 있으며(예: [Fig. 7](b)),

■ 특히 두 번째 대국자인 백(최소 대국자)은 벌어지는 전체 게임에 대한 최소 평균 승률값을 갖는 잘못된 곳을 선택하기 때문에 첫 번째 대국자인 흑(최대 대국자)에게 몇 수내에 게임을 지는 경우가 발생할 수가 있다(예: [Fig. 7](a)).

[Fig. 9]는 MCTS 간의 대국에서 발생될 수 있는 모든 형태인 [Fig. 6]의 3가지 유형에 대한 수순을 보여 주고 있다.



[Fig. 9] All the three types happened during the competition between two MCTS players

실제 10,000번의 게임을 수행하여 첫 번째 대국자인 흑(최대 대국자)의 게임 결과를 분석해 보면 유형 1로 2,342번의 흑 승리, 유형 2로 3,727번의 흑 승리, 그리고 유형 3으로 3,931번의 무승부를 기록하여, [Table 2]에서 보듯이 첫 번째 대국자(흑)가 두 번째 대국자(백)에 대해 80.3±0.8%라는 압도적인 평균 승률을 보였다.

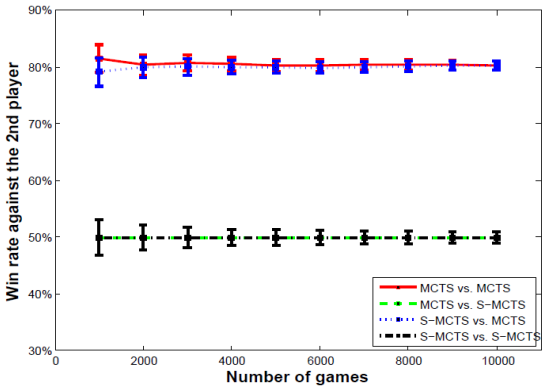
3.3 MCTS의 문제점 개선

MCTS를 활용하여 완벽한 알고리즘을 구현하기 위해서는 앞에서 서술한 문제점을 강제할 수 있는 수단이 필요하다. 즉 전략적 몬테카를로 트리탐색(S-MCTS: Strategic Monte-Carlo Tree Search)을 구축할 필요가 있다. 수학적 확률값이 최대치인 곳 대신에 전략적으로 더 급한 곳을 당장 착수를 안 하여 두 번의 착수 후에 지는 곳을 우선적으로 봉쇄할 필요가 있으며, 구현할 수 있는 S-MCTS의 유사코드는 [Fig. 10]과 같다.

```
function MCTS(Position, NoOfSimulations)
  for each child n available from Position
    reward[n] = 0
    for i from 1 to NoOfSimulations
      POS = copy(Position)
      while (POS is terminal node)
        MCTS(POS, random child from POS)
      end while
      if depth == 2 then
        reward[n] += 2 * result(POS)
      else reward[n] += result(POS)
    end for
    reward[n] /= NoOfSimulations
  end for
  return(child i with highest reward[i])
end function
```

[Fig. 10] Pseudo-code for the S-MCTS algorithm

또한 [Table 2]에서 보듯이 10,000번의 삼목 게임에서 두 번째 대국자인 백으로 S-MCTS가 두는 경우, 첫 번째 대국자인 흑이 MCTS 또는 S-MCTS 여부에 상관없이 50.0±1.0%의 평균 승률을 보여 결코 게임에 지지 않는 것을 알 수 있다. 참고로 [Fig. 11]은 [Table 2]의 네 가지 형태의 대국에서 10,000번의 게임 중 발생된 평균 승률에 대한 추이도가 된다.



[Fig. 11] Performance of MCTS and S-MCTS players playing against each other

[Fig. 11]에서 보듯이 네 가지 형태의 대국은 공히 2,000번 정도의 게임 이후부터 평균 승률이 점차 수렴하고 있음을 알 수 있다. 즉, 두 대국자 모두 MCTS인 경우는 80.3%, 첫 번째 대국자 MCTS와 두 번째 대국자 S-MCTS는 50.0%, 첫 번째 대국자 S-MCTS와 두 번째 대국자 MCTS는 80.3%, 두 대국자 모두 S-MCTS인 경우는 50.0%로 각각 수렴하고 있다.

4. 결론 및 제언

올해 벌어진 알파고와 이세돌과의 바둑 대국은 전 세계적인 관심사가 되었으며, 4:1이라는 압도적인 대국 결과로 인해 인공지능은 이제 4차 산업혁명을 주도할 중요한 과학 분야가 되어 버렸다. 세계 최고의 컴퓨터 바둑인 알파고를 작동시킨 알고리즘 근간에는 MCTS가 있어 그와 같은 위대한 업적을 남길 수가 있었다.

저자는 본 논문을 통해 MCTS의 문제점과 이에 대한 해결책을 제시하고자 했다. 반복적인 무작위 탐험에 의해 생성된 통계적 확률값에 전적으로 의존하는 MCTS가 안고 있는 문제점으로는 (1)수학적 확률값이 같은 경우 이에 대한 적절한 대처가 미흡하며, (2)특히 두 번째 대국자인 최소 대국자는 발생되는 전체 게임에 대한 최소 평균 승률값을 선택하는 행동을 취하기 때문에 첫 번째 대국자인 최대 대국자에게 최선의 대응을 할 수 없는 상황이 벌어지는 경우가 있다. 결국 이러한 문제점을 해결하기 위하여 순수 MCTS가 아닌 전략을 구사할 수 있는 전략적 몬테카를로 트리탐색인 S-MCTS를 본 논문에서 제시하였다. 전략을 구사하는 S-MCTS는 첫 번째 대국자가 MCTS 또는 S-MCTS 여부에 상관없이 항상 최선의 행동을 취해 게임에 결코지지 않음을 실험을 통해 보였다.

결국 본 논문에서의 최대 성과는 향후 MCTS를 활용한 게임프로그래밍을 구축하는 경우, 순수 MCTS가 아닌 전략기반 S-MCTS를 구축해야 한다는 사실을 찾아낸 것이다.

또한 향후 순수 MCTS보다 다소 성능이 우수한 신뢰상한 트리탐색(UCT: Upper Confidence Bounds for Trees)과 전략기반 S-MCTS의 성능을 비교하는 것도 좋은 연구 과제가 될 듯하다.

ACKNOWLEDGMENTS

This paper was supported by the Sehan University Research Fund in 2016.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] D. Silver et al., “Mastering the game of Go with deep neural networks and tree search”, Journal of Nature, Vol. 529, Issue 7587, pp. 484-489, 2016.

[6] P.S. Jang “Overseas Innovation Trend”, from http://www.stepi.re.kr, 2016.

[7] 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.

[8] 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.
[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] G. Chaslot, “Monte-Carlo Tree Search”, Ph.D. dissertation, University of Masstricht, 2010.
[11] Wikipedia, “Computer Go”, from http://en.wikipedia.org/wiki/Computer_Go, 2016.
[12] Wikipedia, “Tic-Tac-Toe”, from http://en.wikipedia.org/wiki/Tic-Tac-Toe, 2016.
[13] 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. http://en.wikipedia.org/wiki/Tic-Tac-Toe, 2016.
[14] 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.
[15] T. Pepels, “Novel Selection Methods for Monte-Carlo Tree Search”, Master thesis, University of Masstricht, 2014.
[16] 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, 2016.
[17] 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, 2016.
[18] B.D. Lee, “Implementation of robust Tic-Tac-Toe game player, using enhanced Monte-Carlo algorithm”, Journal of Korean Society for Computer Game, Vol. 28, No. 3, pp. 135-141, 2015.

이 병 두(Lee, Byung-Doo)

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

관심분야 : 컴퓨터공학, 인공지능, 컴퓨터바둑