턴이 바뀌지 않는 경우가 있는 게임에서의 Monte Carlo Tree Search를 이용한 게임 전략

양호정[O1] 장하영[2] 장병탁[2]

경기과학고등학교[1] 서울대학교 컴퓨터공학부[2]

ghwjd0816@naver.com, hyjang@bi.snu.ac.kr, btzhang@bi.snu.ac.kr

Monte Carlo Tree Search for a board game with nonconsecutive turns

Ho-Jung Yang[O1] Ha-Young Jang[2] Byoung-Tak Zhang[2]

Gyeonggi Science High School for the Gifted[1]

Dept. of Computer Science and Engineering, Seoul National University[2]

요 약

두 명의 사용자가 교대로 게임을 진행하는 많은 보드게임에서 알파베타 가지치기나 미니맥스 알고리즘 등의 트리 알고리즘들을 이용하여 쉽고 효율적으로 게임 전략을 설계하는 것이 가능하다. 그러나 이러한 알고리즘들은 게임의 진행상황에 따라서 규칙적으로 턴이 바뀌지 않는 경우에는 문제 공간의 크기가 너무 커져서 문제의 해결이 쉽지 않고, 말판의 크기가 커짐에 따라서 이러한 문제점은 급격하게 증가하게 된다. 본 연구에서는 이러한 경우에 Monte Carlo Tree Search를 접목하여 탐색에 걸리는 시간을 줄이고, 규칙적으로 턴이 바뀌지 않는 경우로 인해서 발생하는 문제공간의 크기 증가에 따른 문제점들을 처리하기 용이하게 하였다. 제안한 방법론을 사용하여 만들어진 전략과 기존의 트리기반의 알고리즘들을 이용하여 만들어진 전략의 승률을 비교하여 보다 짧은 탐색 시간으로 더 우수한 전략을 만들 수 있음을 확인하였다.

1. 서 론

두 명의 사용자가 교대로 게임을 진행하는 전통적인 보드게임에서 트리기반의 알고리즘들은 쉽고 직관적인 방법으로 상당히 우수한 전략을 만들 수 있는 방법을 제공해왔다.[2][3] 하지만 이러한 트리기반 알고리즘들은 말판의 크기가 증가함에 따라서 문제 공간이 급격하게 커지는 문제로 인하여 작은 규모의 간단한 규칙을 가진 보드게임에 주로 사용되어 왔고, 특히나 본 논문에서 다루고 있는 것 같이 게임의 진행에 따라서 규칙적으로 턴이 바뀌지 않는 보드게임의 경우에는 이러한 문제점들이 더욱 커지게 되므로 기존의 트리기반 알고리즘을 적용하기 어렵게 된다.

본 연구에서 사용되는 몬테카를로 방법은 난수를 이용하여 함수의 값을 확률적으로 계산하는 알고리즘이다. 주로 수학이나 물리학에 사용되어 계산하려는 값이 수학적 형식으로 표현되지 않거나 복잡하여 근사적으로 계산해야 하는 상황에 사용된다. 여러 게임 알고리즘 제작에 사용되는 몬테카를로 방법은 또한 [표 1]에서 볼 수 있듯이 바둑에 적용되어 좋은 결과를 보였다.[1][4]

[표 1] 바둑 프로선수를 상대로 컴퓨터가 처음으로 이긴 게임들의 목록

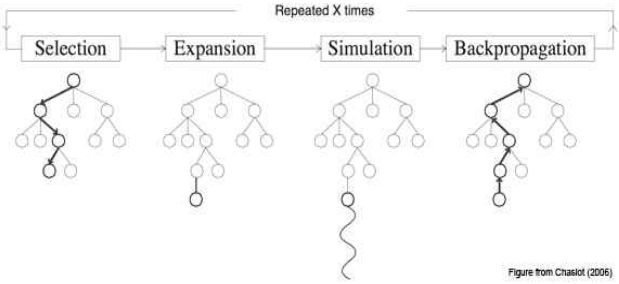
Year	Handicap	Human level	Computer Program
2008	9	8단	MoGo
2008	8	4단	Crazy Stone
2009	7	9단	Mogo
2009	6	1단	Mogo

특히 몬테카를로 방법으로부터 나온 몬테카를로 트리 탐색(Monte-Carlo Tree Search, 이하 MCTS)은 게임에서의 최선의 수를 찾기 위해 유용하게 사용된다.[4] 많은 횟수의 시뮬레이션을 통하여 각 수가 장기적으로 게임에 어떤 영향을 미치는지를 확인한 후에 트리 탐색을 여러 번 진행하면서 노드의 가치가 변화 하는 것을 계산하여 그 수가 좋은 수인지 나쁜 수인지를 판단한다.[2]

본 논문에서는 턴이 바뀌지 않는 경우가 있는 게임에서의 게임전략의 계산에 용이한 MCTS를 사용하여 더 큰 보드판에서도 빠른 속도로 수를 계산하여 둘 수 있도록 하였다. 제안한 방법론의 성능을 확인하기 위하여 다른 트리알고리즘인 미니맥스 알고리즘, 알파베타 가지치기와 탐욕 알고리즘을 사용하여 제작한 전략과의 대결을 통하여 만든 전략의 성능을 확인하였다.

2. 몬테카를로 트리 탐색(Monte-Carlo Tree Algorithm)

MCTS는 최선의 선택(optimal decisions)을 내리기 위한 방법으로 트리에서의 무작위적 시뮬레이션을 통하여 결정한다. MCTS는 임의로 시행되는 다수의 시뮬레이션을 통하여 각각의 움직임을 확인, 장기적으로 효율적인 수를 추정하는데 사용한다. 그 과정은 [그림 1]과 같이 네 단계로 세분화 된다.



[그림 1] Monte-Carlo Tree Search 모식도

선택(Selection)에서는 뿌리 노드에서 시작하여 현재까지 펼쳐진 트리를 선택한다. 확장(Expansion)에서는 만약 그렇게 선택한 트리에서 게임의 종료가 되지 않은 경우에는 하나이상의 자식노드를 생성하여 하나를 선택한다. 시뮬레이션(Simulation)에서는 확장에서 선택된 자식노드에서 게임의 시뮬레이션을 돌려 게임의 종료가 될 때 까지 시행한다. 마지막으로 역전파(Backpropagation)에서 선택된 자식노드에 시뮬레이션 결과를 반영한다. 시뮬레이션 결과에 기초하여 기록된 값은 방문한 횟수와 시뮬레이션의 결과로 나온 점수를 포함해야 한다.[6]

MCTS를 적용하기 위해서는 3가지 조건을 만족해야 하는데 게임의 최대/최소 점수 값이 있어야 하고, 게임 규칙이 정해져 있으며 완전정보 게임이어야 하며 게임의 길이가 제한되어 비교적 빨리 시뮬레이션이 끝나야 한다.[4]

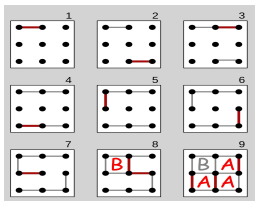
MCTS에서는 위치평가함수가 필요하지 않아 위치평가함수가 필요한 다른 알고리즘들과 달리 턴이 바뀌지 않는 경우가 있는 게임에 대해서 더 간편히 턴이 바뀌지 않는 처리를 해줄 수 있다는 장점을 가지고 있다.

한 부모노드와 자식노드에서의 플레이어가 동일한 경우에 위치평가 함수를 매번 확인하여 적용시켜주어야 하는 반면에, MCTS는 시뮬레이션을 진행할 때 턴이 바뀌지 않는 경우에서의 예외처리만 해주면 되기 때문이다. 또한 재귀적으로 대부분의 수를 두어 결과를 확인해야 하는 미니맥스 알고리즘, 알파베타 가지치기는 탐색의 깊이가 깊어지면 탐색 시간이 O(k^n)인 반면 MCTS는 탐색의 깊이가 길어져도 시뮬레이션 횟수에 비례하여 O(n)의 속도를 내어 판의 크기가 큰 보드게임들에서 유용하게 사용된다.

3. MCTS를 이용한 게임전략

본 연구에서는 Dots and Boxes라는 게임에 MCTS를 접목하여 게임전략을 제작하여 다른 트리알고리즘과 탐욕(Greedy) 알고리즘을 사용한 게임전략과의 대결을 통하여 MCTS를 이용한 게임전략의 성능을 파악하도록 진행하였다.

Dots and Boxes는 정사각형의 격자의 점들 사이를 연결하여 두 플레이어가 번갈아 진행하며 정사각형의 네 변 모두를 연결했을 경우에 득점이 인정되며 한 번의 수를 더 둘 수 있어 본 연구에서 소개한 턴이 바뀌지 않는 경우가 있는 게임의 일종이라고 할 수 있다.[5]



[그림 2] Dots and Boxes

이 연구에서 사용된 MCTS의 의사코드(pseudo code)는 다음과 같다.

```
1  function MCTS(node, player)
2      for simulate_times
3          while game_end
4          generate random move
5          if(first_move) first move <- move
6          if(point)continue
7          else swap player
8          end while
9      backpropagate
10     end for
11     return best move
12 end function
```

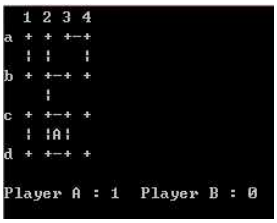
[그림 3] 몬테카를로 트리 탐색 알고리즘의 의사코드

위의 의사코드를 보면 2번 줄에서 시뮬레이션을 주어진 횟수만큼 실행을 시작하여 매번 게임종료 시까지 4~7번의 동작을 반복한다. 4번에서 다음 노드를 선택하는 확장을 진행하고 이때의 노드를 선택하는 것은 무작위로 선택한다. 5번에서는 만약 현재 선택한 노드가 처음으로 선택한 노드일 경우에 시뮬레이션을 시작한 노드가 어느 곳인지 알기 위하여 기록한다. 6번과 7번에서는 진행하는 게임의 특징인 점수를 획득하면 턴이 넘어가지 않으므로 그러한 부분을 적용하였다. 8번에서 게임이 종료가 되었을 경우에 9번에서 역전파를 진행하여 5번에서 기록한 첫 번째 노드에 시뮬레이션 결과와 방문한 횟수를 기록한다. 2번의 for문이 종료될 시 까지 시뮬레이션을 진행하여 11번에서 가장 최선의 수라고 생각되는 노드를 반환한다.

Dots and Boxes에 MCTS를 접목하기 위하여 필요한 과정인 역전파에서 필요한 시뮬레이션 결과에는 두 가지, 득점한 점수와 승리여부가 있다. 득점한 점수는 최종 득점한 점수에서 현재 득점한 점수를 빼어 노드에 더해주었다. 또한 승리여부는 득점한 점수보다 더 중요도가 높게 설정해야 하고 득점한 점수는 판의 크기에 비례하여 증가하므로 승리에 대한 값을 $(n-1)^2+1$(n : 판의크기)에 대응하여 승리 한 경우에는 더해주고 비겼을 경우는 유지, 패배한 경우에는 값을 빼주었다. 또 총 10만 번의 시뮬레이션 반복횟수에 대하여 여러 번 노드가 선택되는 경우에 따라서 매번 평균값을 취해 주어 랜덤 선택에 의하여 좋지 않은 수가 여러 번 선택되는 경우를 배제하여 값을 얻을 수 있도록 하였다.

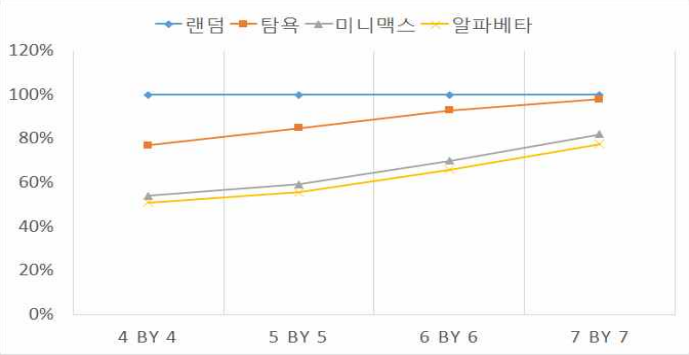
4. 실험결과

승률 비교를 위하여 4개의 전략인 랜덤하게 노드를 두도록 만든 랜덤(Random) 전략, 매번 최대한 실점하지 않을 수 있는 수를 두는 탐욕 알고리즘, 그리고 트리 알고리즘의 일종인 미니맥스 알고리즘과 알파베타 가지치기를 사용하였다. 각 전략 별로 100회 게임을 진행하였으며 4개의 다른 판의 크기인 4 by 4, 5 by 5, 6 by 6, 7 by 7판에 대하여 진행되었다. 또 첫 수를 두는 순서는 번갈아 가면서 50번씩 첫수를 두는 플레이어가 되도록 하였다.



[그림 4] 프로그램 진행 화면

100회 시행동안 MCTS와 다른 전략 사이의 승률을 그래프로 나타내었다.



[그림 5] MCTS를 이용한 게임전략의 판의 크기에 따른 상대 알고리즘 별 승률 그래프

랜덤 전략에 대해서는 모든 판에서 100%의 승률을 가졌고 탐욕 알고리즘에 대해서는 판의 크기가 커짐에 따라 각각 77%, 84.8%, 93%, 98.0%, 미니맥스 알고리즘에 대해서는 54%, 59.2%, 70%, 82%, 그리고 마지막으로 알파베타 가지치기에 대해서는 51%, 55.6%, 66%, 77.6%의 승률을 보였다.

모든 전략에 대하여 문제 공간의 크기가 커질수록 MCTS를 사용한 전략의 승률이 높았으며 특히 트리 알고리즘의 일종인 미니맥스 알고리즘과 알파베타 가지치기에 대해선 4 by 4일 때와 7 by 7일 때의 차이가 각각 28%, 25.6%로 큰 차이를 보였다. 이러한 사실로부터 문제공간의 크기가 커짐에 따라서 제안한 방법론이 기존의 트리기반 알고리즘들보다 효율적으로 문제공간을 탐색하여 최적 해를 찾아낸다는 것을 확인할 수 있었다.

판이 커짐에 따라 승률의 증가하는 추세는 탐욕 알고리즘에 대해서는 판의 총 점수가 증가함에 따라 점수를 최대한 잃지 않고 점수를 가져올 수 있을 때 가져오는 것이 아닌 지금 점수를 주면서 후에 더 큰 점수를 가져오도록 하는 전략이 더 필요했던 것으로 보인다.

미니맥스 알고리즘이나 알파베타 가지치기에 대해서는 승률이 4 by 4 판이나 5 by 5 판에서는 탐색 깊이가 그다지 깊지 않아서 MCTS와 비슷한 성능을 내어 50% 근처의 승률을 가질 수 있었던 반면에 6 by 6, 7 by 7로 판이 커지게 되는 경우는 탐색 깊이가 깊어져 미니맥스나 알파베타 가지치기로 탐색할 수 있는 깊이가 한정적이 되어 모든 깊이를 다 돌 수 있는 MCTS의 성능이 월등히 높아진 것으로 보인다. 4 by 4와 5 by 5에서 탐색 깊이가 같은데 성능에서 차이가 난 부분은 미니맥스 알고리즘과 알파베타 가지치기를 이용한 게임 전략에서 승리의 보상에 비해 점수의 보상이 커서 승리를 위한 수가 아닌 득점을 위한 수 역시 두었을 경향이 있던 것으로 보인다.

5. 향후 연구 및 결론

본 논문에서는 트리 알고리즘의 일종인 MCTS를 이용하여 Dots and Boxes라는 턴이 넘어가지 않는 경우가 있는 게임의 전략을 제작해 보았다. MCTS는 다른 트리 알고리즘(미니맥스 알고리즘, 알파베타 가지치기)등과 달리 위치평가 함수가 필요하지 않아 턴이 넘어가지 않는 경우가 있는 게임에 대해서 턴이 바뀌지 않는 경우에 대한 예외 처리에 용이하다는 장점이 있다.

제작한 게임전략의 성능을 알아보기 위해 4가지 알고리즘(랜덤, 탐욕 알고리즘, 미니맥스 알고리즘, 알파베타 가지치기)와 100회 게임을 진행하여 승률을 비교 해보았다. 그 결과 전체적으로 승률이 높게 나왔으며 특히 판의 크기가 증가할수록 승률이 증가하는 추세를 보이는데 이는 판의 크기가 증가함에 따라 탐색해야할 깊이가 깊어져 미니맥스 알고리즘이나 알파베타 가지치기보다 탐색 깊이와 실행시간이 영향이 비교적 적은 MCTS를 사용한 게임전략의 승률이 높게 나온 것으로 보인다.

턴이 바뀌지 않는 경우가 있는 보드게임에서 MCTS를 사용하여 제작한 게임전략을 다른 알고리즘들을 사용한 게임전략에 비교하여 승률을 확인하였을 때, 그 승률이 높은 것에 근거하여 MCTS가 턴이 바뀌지 않는 경우가 있는 보드게임의 게임전략 제작에 유용하다는 결론을 낼 수 있었다.

본 연구에서는 선택 단계를 진행할 때 순수하게 게임 종료 시까지 무작위 선택을 하는 기본적인 MCTS만 사용하였다. 그렇게 되는 경우에 탐색 깊이는 깊어지지만 탐색 속도는 느려지게 된다. 이에 대한 해결 방안으로 UCT(Upper Confidence Bound 1 applied to trees)가 적용되어 탐색 속도와 탐색 깊이의 균형을 맞추는 방안이 제안되었다.[7] 이를 적용하면 더 큰 판(8 by 8, 9 by 9)에서도 빠른 속도를 가지는 게임전략을 제작할 수 있을 것 같아 이러한 부분에서 더 발전을 시킬 수 있을 것으로 보인다.

참고문헌

[1]이병두. “삼목 게임에 적용된 몬테카를로 트리탐색”, *Journal of Korea Game Society 2014 Jun;14(3)*, p47~p54, 2014

[2]박현수, 김경중. "게임 인공지능 최신 연구 동향." *정보과학회지*. p12~p13. 2013

[3]Powley, E. J., Daniel Whitehouse, and P. I. Cowling. "Determinization in Monte-Carlo tree search for the card game Dou Di Zhu." *Proc. Artif. Intell. Simul. Behav., York, United Kingdom*. p17~p24. 2011

[4]Chaslot, Guillaume, et al. "Monte-Carlo Tree Search: A New Framework for Game AI." *AIIDE*. 2008.

[5]Li, Shuqin, Dongming Li, and Xiaohua Yuan. "Research and Implementation of Dots-and-Boxes Game System." *Journal of Software 7.2*. p256~p262. 2012

[6]Chaslot, Guillaume. “Monte-carlo tree search”. *Research Report, Netherlands Organisation for Scientific Research*. p110. 2010.

[7]Kocsis, Levente, and Csaba Szepesvári. "Bandit based monte-carlo planning." *Machine Learning: ECML 2006. Springer Berlin Heidelberg*. p282~p293. 2006