Artifitial Intelligence: Final Project

Gomoku

Project Report

Chen Liguo, Wenqian Zhang
School of Data Science
Fudan University
Shanghai, China
reeddotaer@outlook.com,
zhangwq16@fudan.edu.cn

In this project, we will discuss how we implement MCTS algorithm to solve the Gomoku game. Besides, we will also demonstrate how we promote the MCTS algorithm with ADP and compare the results of MCTS with minimax algorithm we submitted in the midterm report.

Keywords: MCT, ADP, heuristic function

I. Introduction

Monte Carlo Tree Search (MCTS) has been widely used in board games as well as turn-based strategy video games. The starting point behind MCTS is naïve. If we can enumerate all the possible situations after one move, we can certainly pick up the best move. However, in practice, the number of possible situations on a board is too large to enumerate. It would be way too inefficient computationally. MCTS adopts UCT (Upper Confidence Bound1 for Trees) and simulation to address this issue. The whole process of MCTS in one turn is composed of four steps: selection, expansion, simulation and backpropagation. We will discuss how the four steps work in detail in following sections.

Considering the restriction of time and space complexity, we take some adaptations of the MCTS method and apply a new designed heuristic function to evaluation the value of the board. Furthermore, we also combine MCTS method and ADP together, and use a three-layers- Perceptron to evaluate the board. We trained the ADP use a shallow forward neural network, while its performance still need to improve.

In section 2, we will summarize the minimax algorithm with alpha-beta pruning, which we submitted in the midterm report.

In section 3, we will introduce the MCTS algorithm and MCTS-ADP algorithm. We will also demonstrate how we design the heuristic function for the MCTS-ADP algorithm.

In section 4, we will compare the result of all algorithms we employed, including alpha-beta pruning, MCTS, ADP, MCTS-heuristic, MCTS-ADP.

II. Monte Carlo Tree Search

Selection of MCTS

In the selection step, we can use Upper Confidence Tree method proposed by Kocsis and Szepesvári as our policy, which relying on Upper Confidence Bound(UCB) Value to make a trade-off of exploration and exploitation.

$$UCB = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$$

- w_i stands for the number of winning.
- n_i stands for the number of simulations for this specific node.
- N_i stands for the total number of simulations.
- c is the exploration parameter – theoretically equals to $\sqrt{2}$.

UCT is originated from HMCTS, but it is more effective and complexed than HMCTS. UCB can help to find out the balanced leaf nodes earlier than simple HMCTS algorithm, which means that the UCT can reduce time and space complexity that the original version. In fact, UCB keeps great balance between exploration and exploitation. If one node is visited few times or never, its n_i will be very small but its UCB value will be very large, which reminds our agent to improve the priority of this node.

Expansion of MCTS

When visiting a leaf node which has been visited before, we will expand the child nodes of this leaf node. In our algorithm, we choose one proximity from on-board pieces as the child

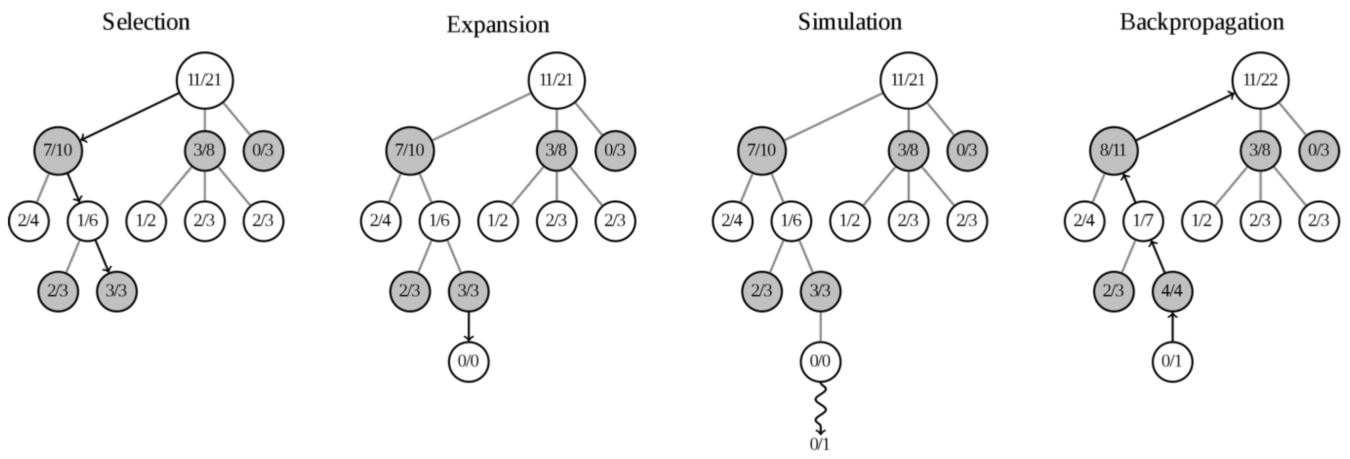

Figure 1: The four steps of MCTS in one turn. The value on nodes is the UCB value. Selection step picks up the node with largest UCB value. Expansion step will expand a leaf node. Simulation step will do simulations according to current situation until the game ends. Backpropagation step will backpropagate the result of simulation and increment the visit times of every parent node.

nodes, i.e. $x \pm 1$ *or* $y \pm 1$, (x, y) is the coordination of on-board nodes.

Simulation of MCTS

The Monte Carlo property of MCTS is the simulation part. When we visited one node the very first time, we will do simulations from this node. The vanilla MCTS randomly choose places from all possible places, reverse the player and repeat until it's terminated. More advanced MCTS will use heuristic function or neural network to evaluate current node.

Backpropagation of MCTS

After each simulation, the number of winning and visiting will be updated from current node to its parent nodes by backpropagation.

MCTS-ADP Exponential Heuristic Algorithms

As a novel method we also consider the Reinforcement Learning in this project. We found that Dongbin Zhao applied Adaptive Dynamic Programming method based on neural network to learn to play Gomoku previously. They train AI by playing games with itself. Then Zhengtao Tang combined Monte Carlo Tree Search with ADP. They use ADP train shallow neural network and calculate the weighted sum of ADP winning probability of simulation in MCTS.

A. Heuristic Function

To reducee the computation complexity, we are looking for an efficient and practical heuristic function. This heuristic function is used to evaluate the rewards of the next action which includes both offense rewards and defense rewards.

Inspired by the previous work we dicide to implement Monte Carlo Tree Search with Adaptive Dynamic Programing in the Gomoku game. Furthermore, we redesign some parts in the program to get the improvement of effect and effeciency.

Heuristic 1: With reference to the previous work, we found that Jun Hwan Kang and Hang Joon Kim proposed a fomula as below.

$$H_i = \sum_l \{(L_{open})^2 + \left(\frac{L_{hclose}}{2}\right)^2\}$$

The variable L_{open} is the length of a line without opponent piece at either ends. The variable L_{hclose} is length of a line with only one opponent's piece at one of the end.

While we find that this formula has some drawbacks. We think it ingores some protential patterns which is discontinuous. Furthermore, the heuristic evaluation value is not reasonable in our opinion. For example, the pattern $L_{open} = 3$ and $L_{hclose} = 4$ should have similar offense effect while they are differ in the heuristic funciton. So we propose a new heuristic function as below.

1) ***Heuristic 2:*** *With respect to the fact that longer patterns are much more crutial in most of the cases, we designed a exponential heuristic function as below.*

$$H_i = \sum_l \{x^{L_{open}} * \alpha^j + x^{L_{hclose}-1} * \alpha^k\}$$

The definition of variable L_{open} and L_{hclose} is as above. The constant x is the base, as in our program we choose it as 10.

And the parameter α is a descend factor which is set as 0.85 in our program.

Possible position function

As we consider the heuristic value of discontinuous patterns we need also to reconsider the function which selects the possible positions of the next step.

If we choose the former one-hop possible positions as our candidates set we might loss some chances as there are still some two-hop positions which might construct new discontinuous patterns.

So we designed our function which will select the two-hop possible positions now.

B. ADP

We also implement a 3-Layer Perceptron to evaluate the value of board. We designed the input features combining all the 29 patterns in our Heuristic 2 and a one hot vector representing whose turn in the next movement. The number of patterns are encoded is shown in table below, where the number is represented by a vector of size 5.

Value of n	Input 1	Input 2	Input 3	Input 4	Input 5
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
>4	1	1	1	1	$\frac{n-4}{2}$

And the activation function of MLP is the sigmoid function:

$$g(x) = \frac{1}{1+e^{-x}}$$

The sigmoid function normalizes the output into (0,1). We use it to transform the output of MLP to winning rate.
We train the MLP in an Adaptive Dynamic Programming process. Only when the AI wins, it get a reward 1, in the other time, the reward is set to 0.

III. Experiment Results Comparison and Analysis

To compare the performance of different

Agent	Mushroom	PureProcky	Valkyrie	Pisq7
Alpha-beta Pruning	12:0	8:4	8:4	2:10
MCTS	2:10	0:12	0:12	0:12
ADP	8:4	6:6	2:10	1:11
MCTS-Heuristic	12:0	9:3	8:4	1:11
MCTS-ADP	12:0	11:1	10:2	2:10

Firstly, we can see that the MCTS method with UCT has poor performance comparing to the other method. This is caused by the limit of time and space restriction and the simulation results have to be truncated early and the simulation value cannot represent the real winning rate. And the adaptation of MCTS all have significant improvement.

The version of MCTS with heuristic function can get good result as it can treat some crucial situation properly when there is chance to construct a good offense and defense pattern.

While the ADP version performs not as well as we think, the reason might be the train is not enough.

While the combination of the two versions works better than any one alone.

With well-designed evaluation function, the alpha-beta pruning method can also get good grades, in the previous work, we can continue to optimize the evaluation function. And the efficiency of alpha-beta pruning method still need to improve for deeper search depth and larger branching factor, which will improve the performance of our program. As in this period of time we devout ourselves into exploring new methods and do not have enough time on the alpha-beta pruning method.

IV. Reference

[1] Browne C B , Powley E , Whitehouse D , et al. A Survey of Monte Carlo Tree Search Methods[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2012, 4(1):1-43.
[2] Tang, Zhentao & Zhao, Dongbin & Shao, Kun & Lv, Le. (2016). ADP with MCTS algorithm for Gomoku. 1-7. 10.1109/SSCI.2016.7849371.
[3] Zhao D, Zhang Z, Dai Y. Self-teaching adaptive dynamic programming for Gomoku[J]. Neurocomputing, 2012, 78(1):23-29.
[4] Kang, J.H.. (2016). Effective Monte-Carlo tree search strategies for Gomoku AI. 9. 4833-4841.
[5] Tan K L , Tan C H , Tan K C , et al. Adaptive game AI for Gomoku[C]. 2009 4th International Conference on Autonomous Robots and Agents. IEEE, 2009.
[6] Cowling P I , Powley E J , Whitehouse D . Information Set Monte Carlo Tree Search[J]. IEEE Transactions on Computational Intelligence & Ai in Games, 2012, 4(2):120-143.
[7] Chen C H , Lin S S , Chen Y C . An algorithmic design and implementation of outer-open gomoku[C]// 2017 2nd International Conference on Computer and Communication Systems (ICCCS). IEEE, 2017.