Report on Assignment 3

Dingran Qi
Vrije Universiteit Amsterdam
2761884
d.qi@student.vu.nl

Tianzheng Hu
Vrije Universiteit Amsterdam
2760270
t.hu2@student.vu.nl

1. PROBLEM EXPLANATION

This is a problem about how to maximize the probability of winning a tic-tac-toe game. The opponent who playing cross has already made the first move in the middle, and its each subsequent move is chosen at random. The player playing circle needs to determine the next move with the highest probability of winning according to the Monte Carlo tree search.

2. ALGORITHM DESCRIPTION

2.1 Terminal condition

The game-over condition is first defined. In Tic-Tac-Toe, the player with the same pattern wins when there are three identical patterns in any row, column, or diagonal. In addition, a game is considered a draw when all nine spaces are filled, but none of the above conditions occurs. The reward is 1 for winning and 0 for losing and drawing.

2.2 Monte Carlo tree search

The root node is first generated based on the initial board. Then its child nodes are generated based on the free positions of the root node, and a child node is randomly selected for expansion. Circle and Cross cycle through this process until the end of the game. When the game is over, the score is determined based on the state of the board. From the leaf node back to the root node, the number of visits to each node on this path is added by 1. If the circle wins at a leaf node, the winning number of each node on the path also increases by 1.

In a purely random Monte Carlo tree search process, both cross and circle move randomly. Nevertheless, in order to reduce useless exploration, the UCB rule is adopted to select the child nodes to explore when it is the circle's turn to move and choose the child node with the highest UCB. The UCB value can be calculated as following:

$$\frac{w_i}{n_i} + c\sqrt{\frac{lnN_i}{n_i}} \tag{1}$$

In this formula, w_i is the number of wins for the node after i times move. n_i is the number of simulations for its child node. N_i is the total number of simulations for the parent node. c is an exploration parameter which usually be $\sqrt{2}$.

Theoretically, the process of the Monte Carlo tree search whit the UCB rule is following:

step 1. Generate the Mento Carlo tree with the root node.
step 2. (Selection. If the play turn is a circle, choose a child node with the highest UCB value to explore. If the play turn is cross, choose a child node randomly. Repeat this step till the leaf node.
step 3. (Expansion. If the game is not over, add child nodes and choose one node randomly based on the current node board.
step 4. (Simulation. Perform a rollout for the leaf node generated in step 3. Choose the child node randomly till the game is over.
step 5. (Backtrack. Determine the score by the previous terminal conditions on the leaf nodes. Update the value of win and visit for every node chosen in step 2 and step 3.

2.3 Convergence process

In order to control the number of iterations of the Mento Carlo tree searching, the Q-value is used as a judgment condition. When the node is a leaf node, which means the winner has been determined. In this case, the Q-value can simply be calculated as the probability of winning. By dividing the number of wins and the number of visits:

$$Q(x) = \frac{w_i}{n_i} \tag{2}$$

If the node is not a leaf node, in order to get the Q-value of this node, we have to recurse the child nodes. $n(x)$ is the number of visits to this node. $n(x')$ and $Q(x')$ are the numbers of visits and Q-value of the child node respectively.

$$Q(x) = \frac{\sum_{x'} n(x')Q(x')}{n(x)} \tag{3}$$

The search for a Monte Carlo tree is considered to have converged to the optimal policy when the Q-value no longer changes. In order to quantify "no longer changes", set a small number ϵ as bound. When the Q-value's difference of two iterations is less than ϵ, stop interning.

$$Q(x) - Q(x') < \epsilon \tag{4}$$

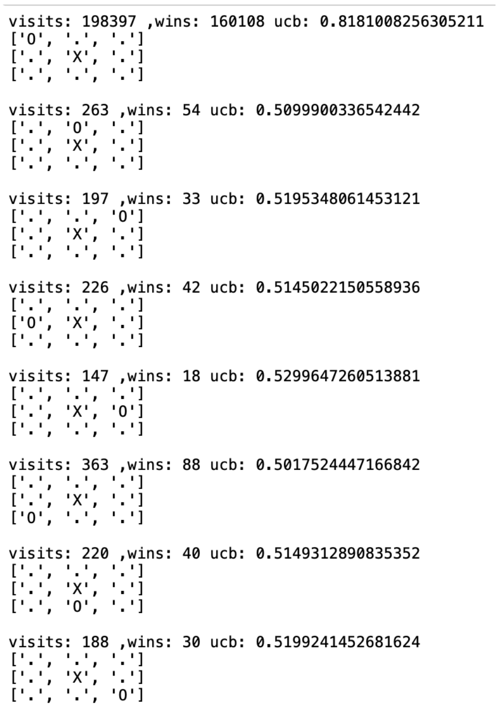
$$Q(x) - Q(x') < 0.000001 \tag{5}$$

3. RESULT

3.1 Mento Carlo tree search with UCB

When the Monte Carlo tree is first created, the number of visits and wins of all nodes are 0, which means the UCB value is an infinite number. As the Monte Carlo tree searching processes, the UCB value is updated to smaller number depends on the number of visits and wins. The UCB rule allows programming gives up some particular branches with bad rewards but favour those with good rewards.

In this tree, the print results show the node that placing circle in the left top corner of the board has the highest UCB value than the others in the first circle's turn(1).



```
visits: 198397 ,wins: 160108 ucb: 0.8181008256305211
['O', '.', '.']
['.', 'X', '.']
['.', '.', '.']

visits: 263 ,wins: 54 ucb: 0.5099900336542442
['.', 'O', '.']
['.', 'X', '.']
['.', '.', '.']

visits: 197 ,wins: 33 ucb: 0.5195348061453121
['.', '.', 'O']
['.', 'X', '.']
['.', '.', '.']

visits: 226 ,wins: 42 ucb: 0.5145022150558936
['.', '.', '.']
['O', 'X', '.']
['.', '.', '.']

visits: 147 ,wins: 18 ucb: 0.5299647260513881
['.', '.', '.']
['.', 'X', 'O']
['.', '.', '.']

visits: 363 ,wins: 88 ucb: 0.5017524447166842
['.', '.', '.']
['.', 'X', '.']
['O', '.', '.']

visits: 220 ,wins: 40 ucb: 0.5149312890835352
['.', '.', '.']
['.', 'X', '.']
['.', 'O', '.']

visits: 188 ,wins: 30 ucb: 0.5199241452681624
['.', '.', '.']
['.', 'X', '.']
['.', '.', 'O']
```

Figure 1: 8 child nodes of root node with visits and wins and ucb value

3.2 Q-value

After 200001 iterations, the Q-value converged to 0.8. The result shows that the Q-value increases rapidly at the beginning, then it gradually slows down the raising ascent (2). Since the difference of two-time iterations is less than the ϵ we set (0.000001), we suppose the optimal policy according to the highest Q-value has been found.

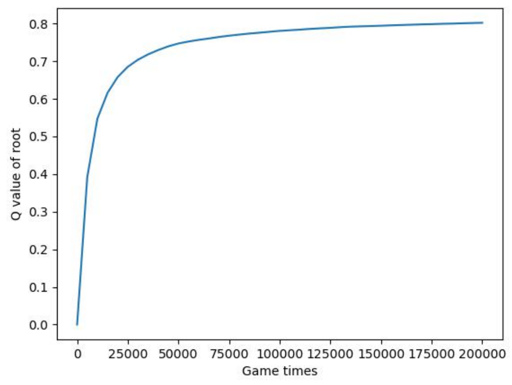
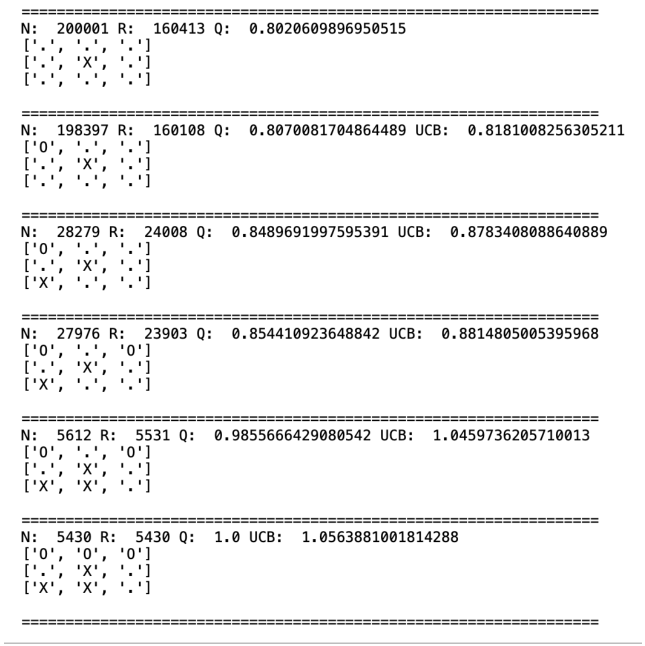


Figure 2: Changing process of Q-value

4. GAME VISUALIZATION

The visualization of one game (3). The circle was placed in the left-top corner in the first round. Then the cross was chosen to place in the left-down corner. In the second, the circle was also chosen to place in a corner place in the top-right. Then the cross was chosen to place in the middle of the third line. In the last round, the circle won the game by placing in the middle of the top line.



```
=================================================================
N:  200001 R:  160413 Q:  0.8020609896950515
['.', '.', '.']
['.', 'X', '.']
['.', '.', '.']

=================================================================
N:  198397 R:  160108 Q:  0.8070081704864489 UCB:  0.8181008256305211
['O', '.', '.']
['.', 'X', '.']
['.', '.', '.']

=================================================================
N:  28279 R:  24008 Q:  0.8489691997595391 UCB:  0.8783408088640889
['O', '.', '.']
['.', 'X', '.']
['X', '.', '.']

=================================================================
N:  27976 R:  23903 Q:  0.854410923648842 UCB:  0.8814805005395968
['O', '.', 'O']
['.', 'X', '.']
['X', '.', '.']

=================================================================
N:  5612 R:  5531 Q:  0.9855666429080542 UCB:  1.0459736205710013
['O', '.', 'O']
['.', 'X', '.']
['X', 'X', '.']

=================================================================
N:  5430 R:  5430 Q:  1.0 UCB:  1.0563881001814288
['O', 'O', 'O']
['.', 'X', '.']
['X', 'X', '.']

=================================================================
```

Figure 3: Visualization of one game